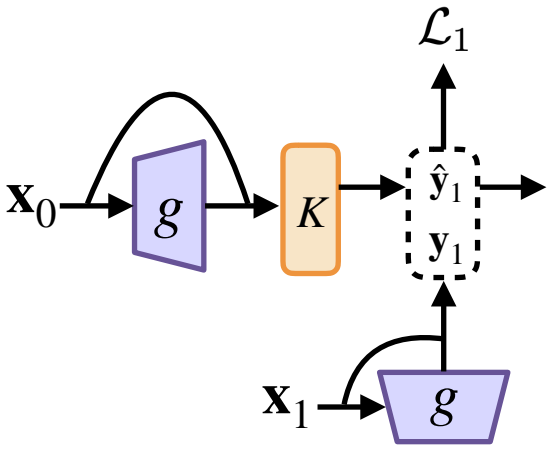
UNIVERSITY OF CAMBRIDGE

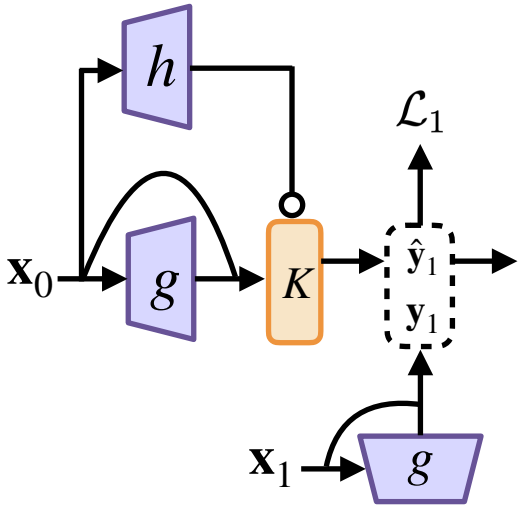
Project Summary W1-2

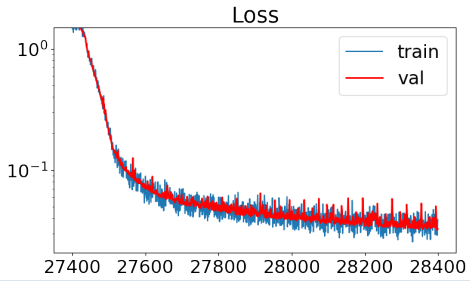
Recap: Two Different NNs

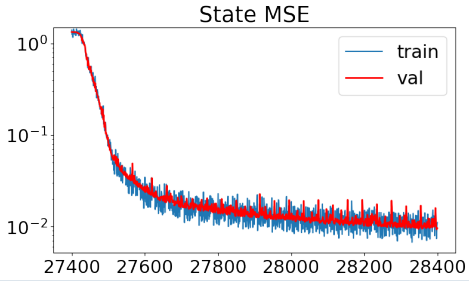
LREN:
Linearly **R**ecurrent **E**ncoder **N**etwork

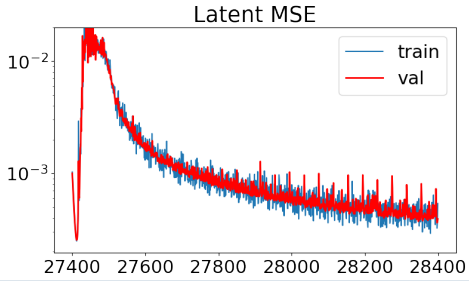


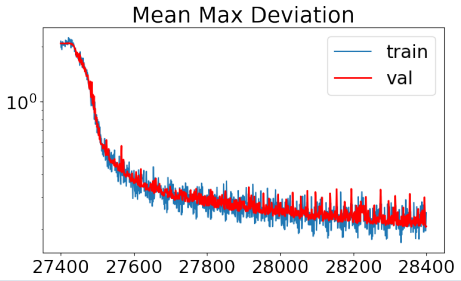
DENIS:
Deep **E**ncoder **N**etwork with
Initial **S**tate Parameterisation





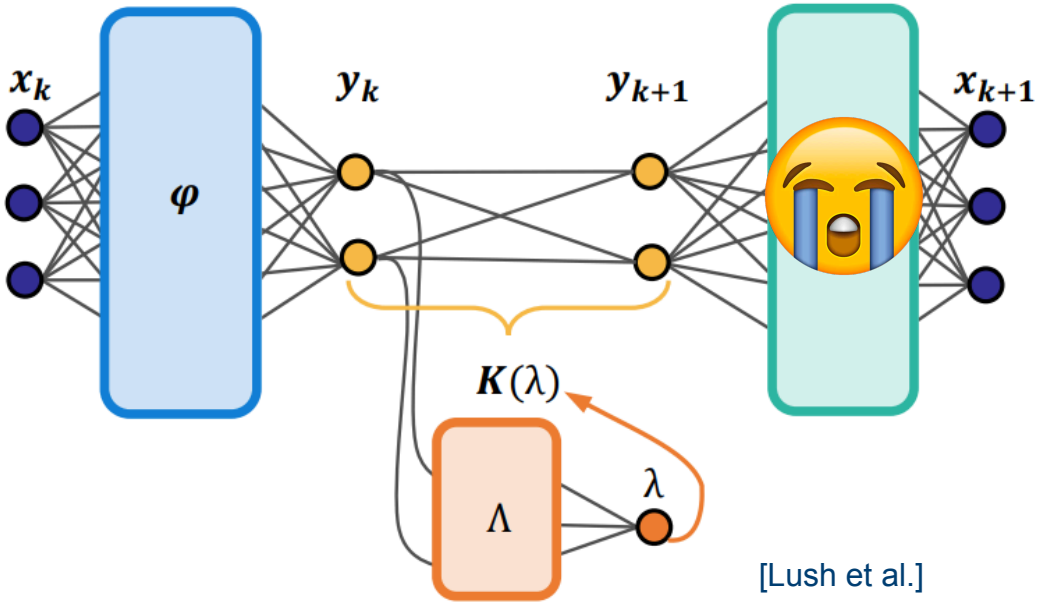






Side Note

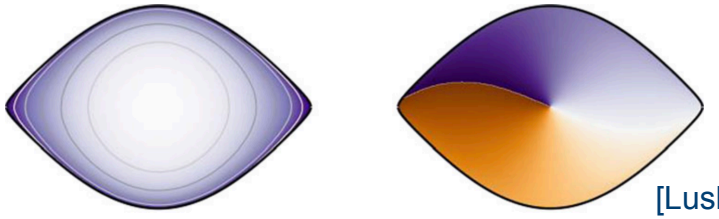
Ideally, only 2 latent layers $Dim(y) >> 2$



[Lush et al.]

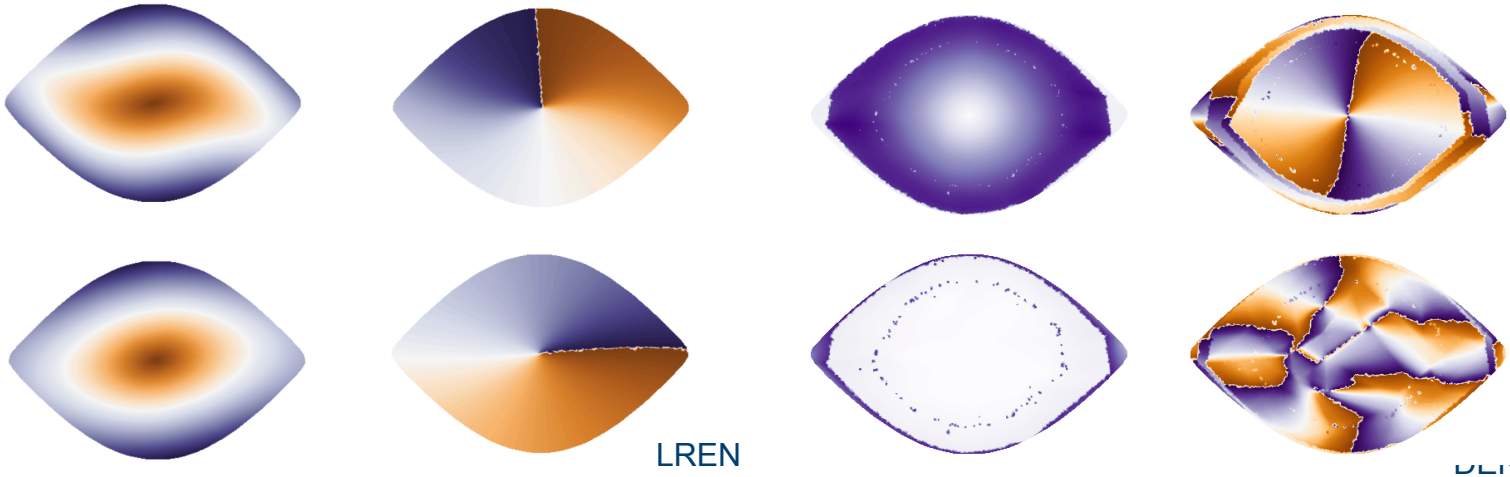
DENIS VS LREN Showdown

Comparison of eigenfunctions:



[Lush et al.]

$$E = m\frac{\left(l\dot{\theta}\right)^2}{2} + mgl(1-\cos(\theta))$$

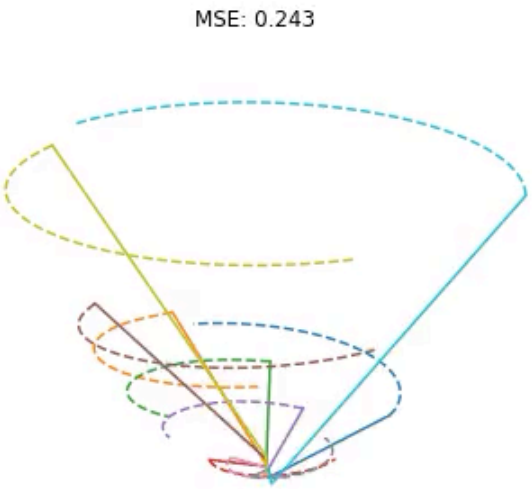


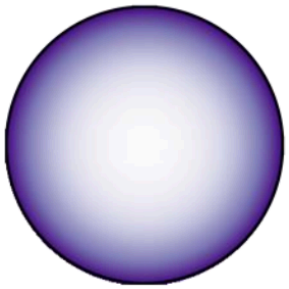
LREN

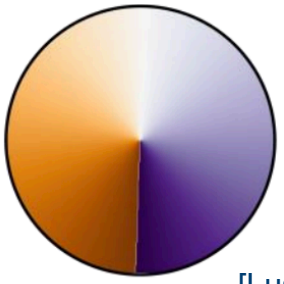
DENIS

Side Note: DENIS Fluid Example

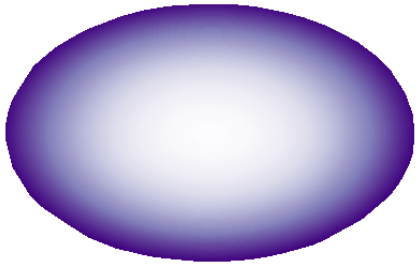
Comparison of eigenfunctions:

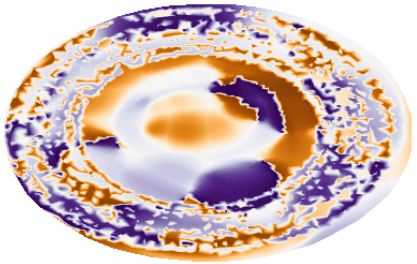






[Lush et al.]

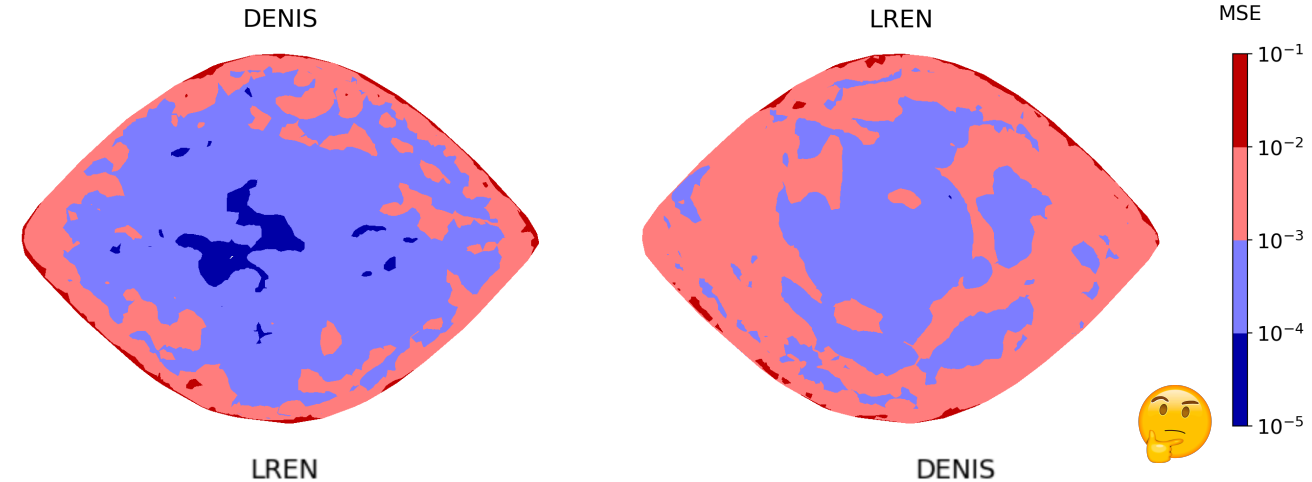


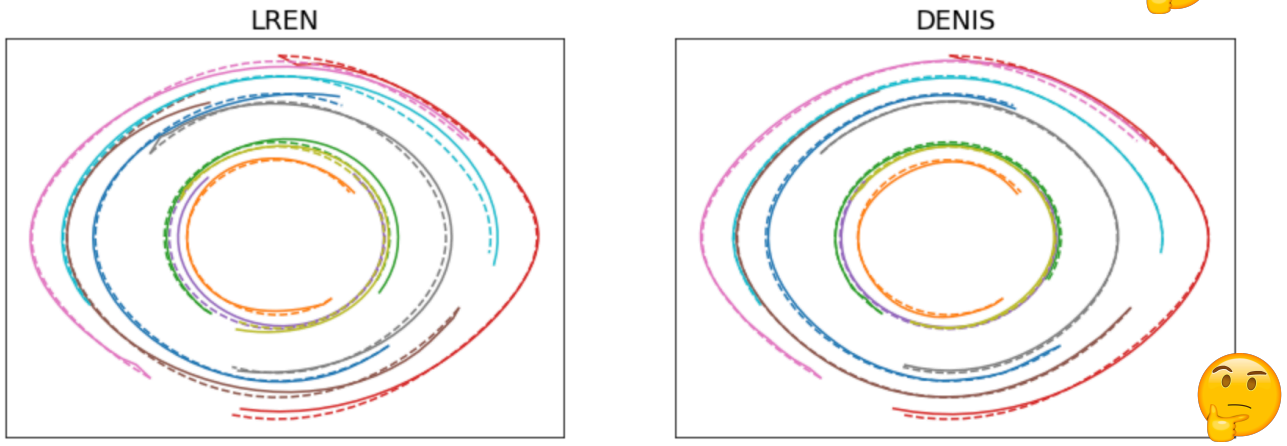


DENIS

DENIS VS LREN Showdown

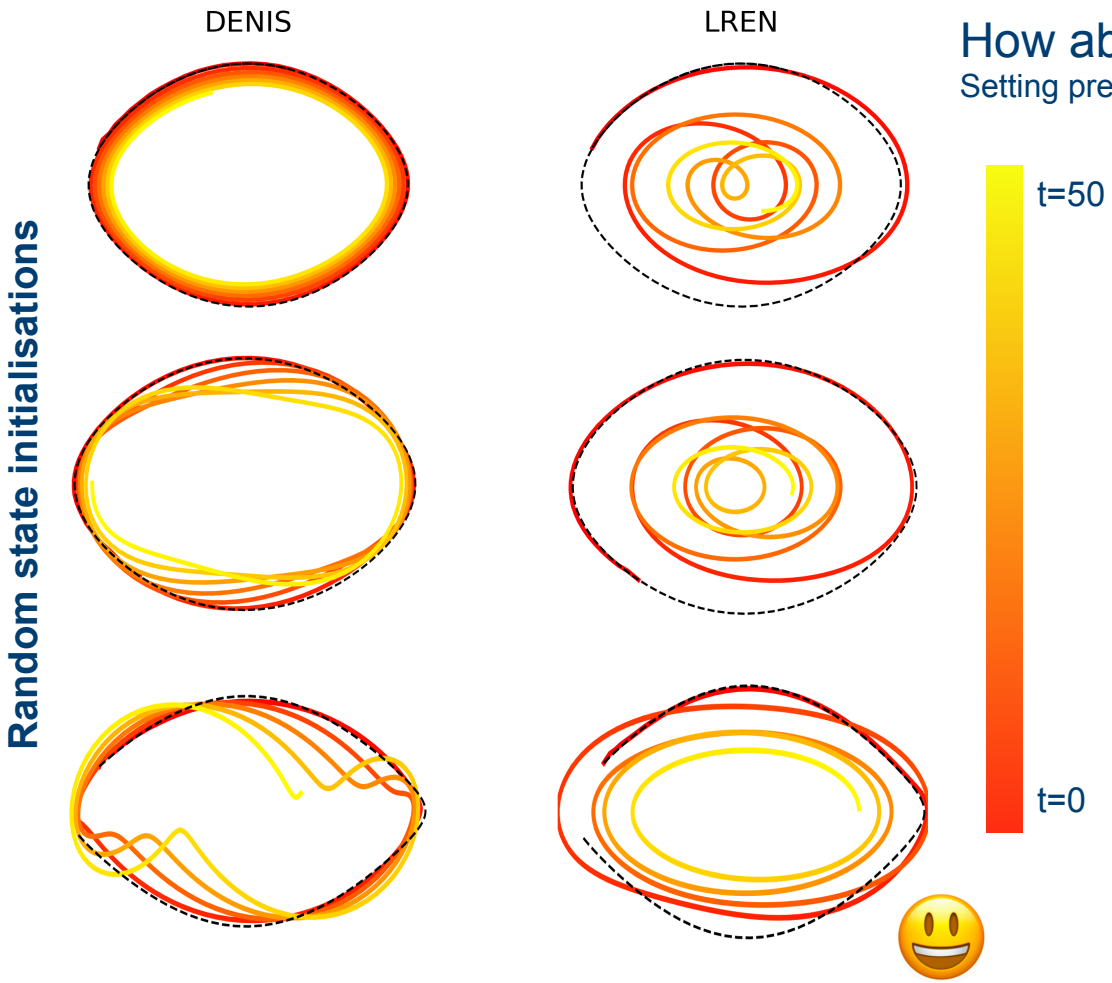
Do we get better predictions?





Average prediction error across all 5 seconds

DENIS VS LREN Showdown



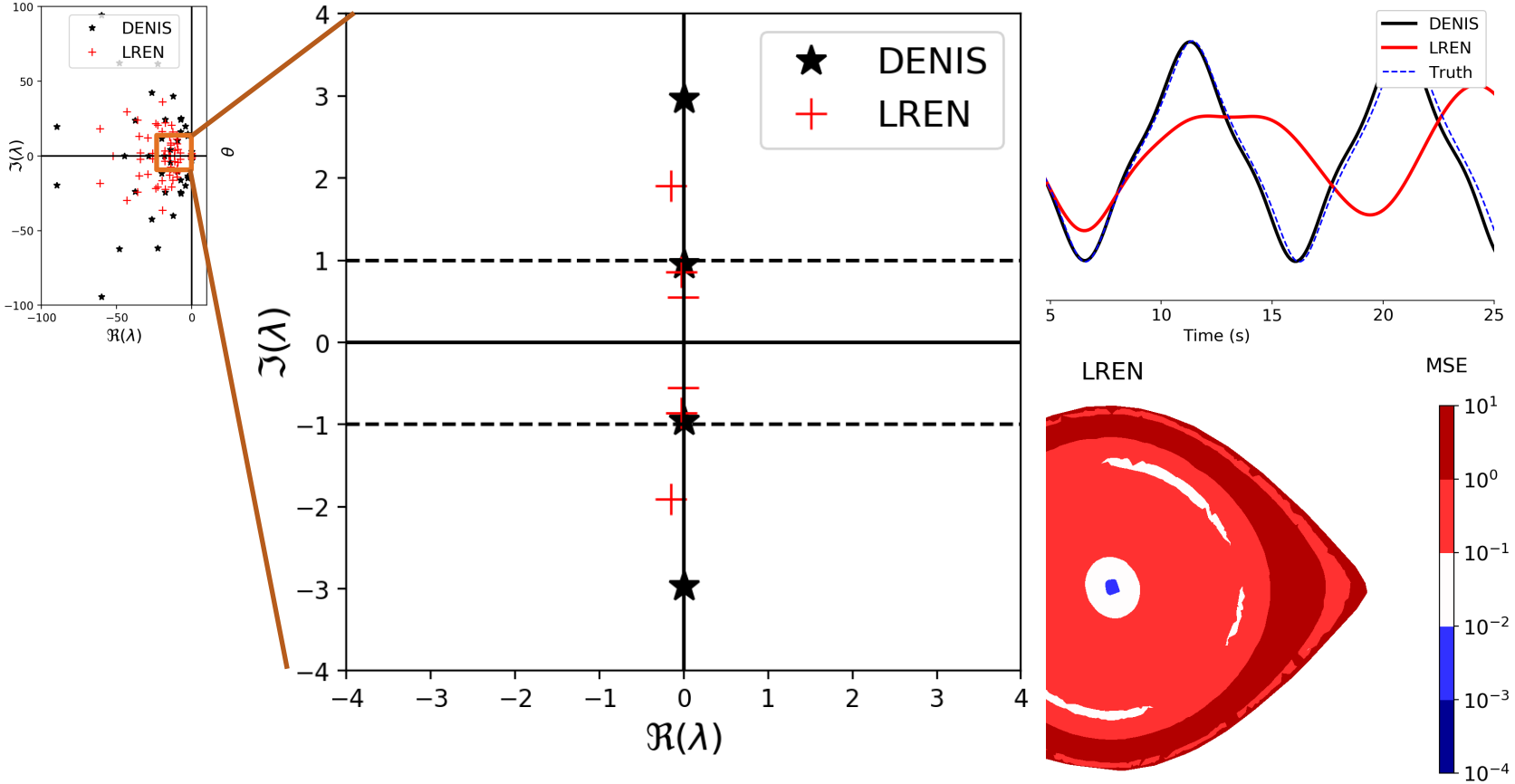
How about long time-horizon predictions?

Setting prediction horizon to 50 seconds (500 steps!)

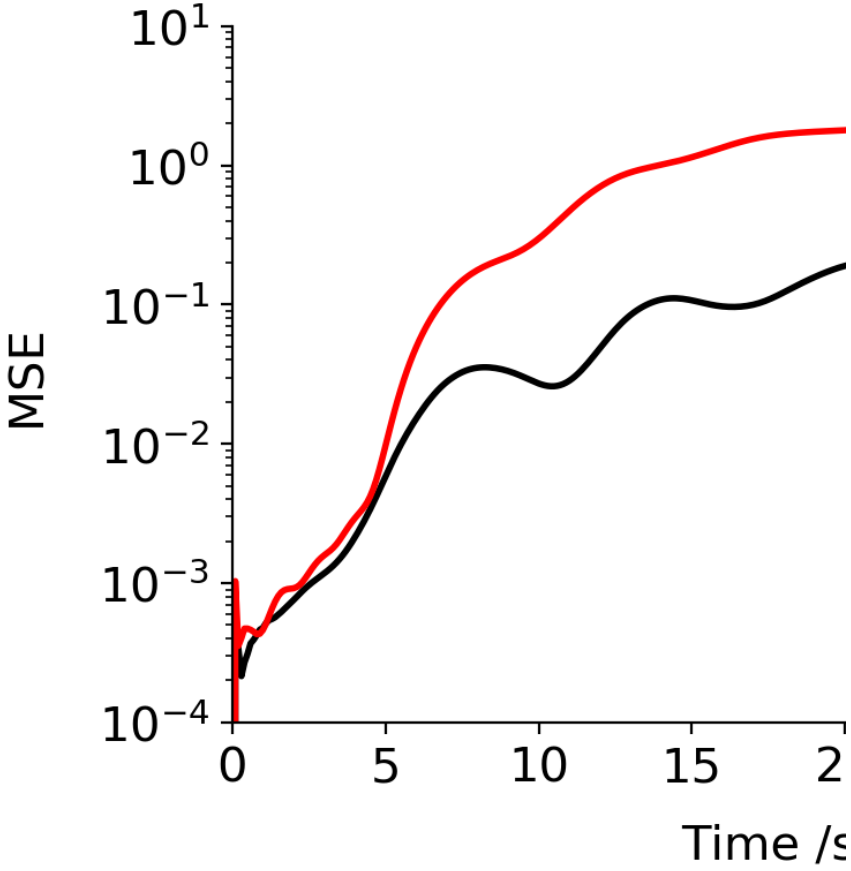
DENIS VS LREN Showdown

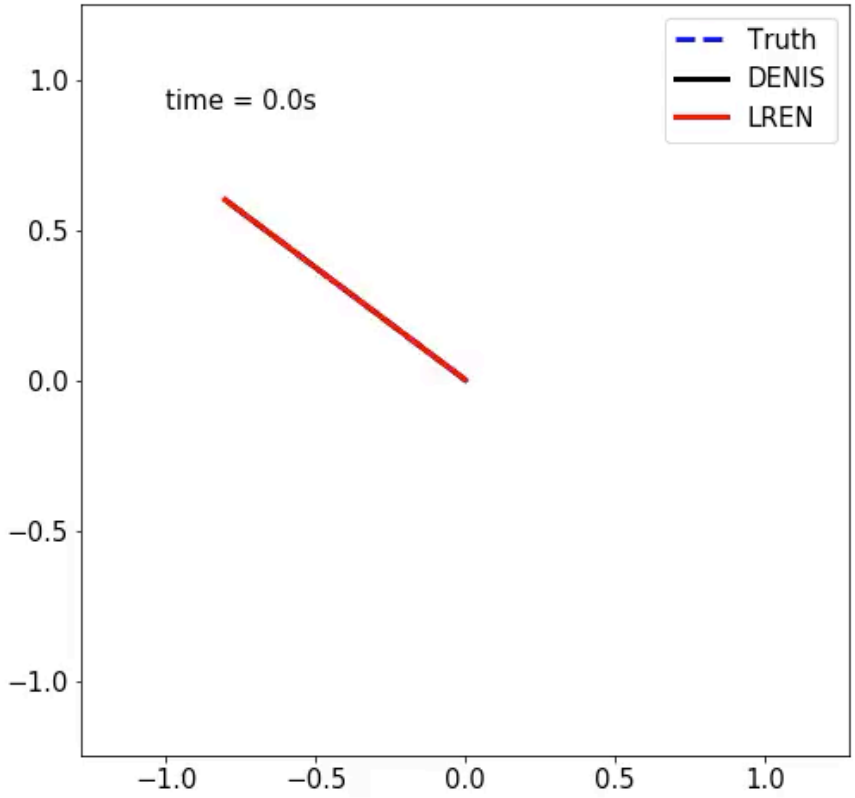
How about long time-horizon predictions?

Setting prediction horizon to 50 seconds (500 steps!)



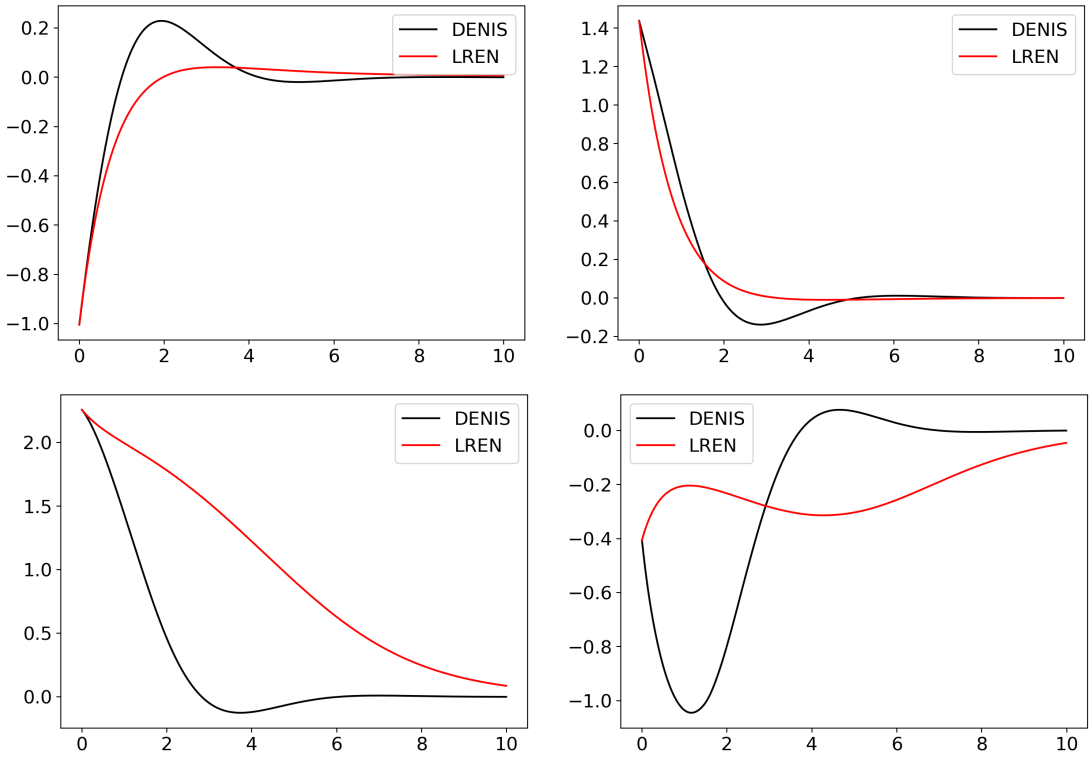
DENIS VS LREN Showdown





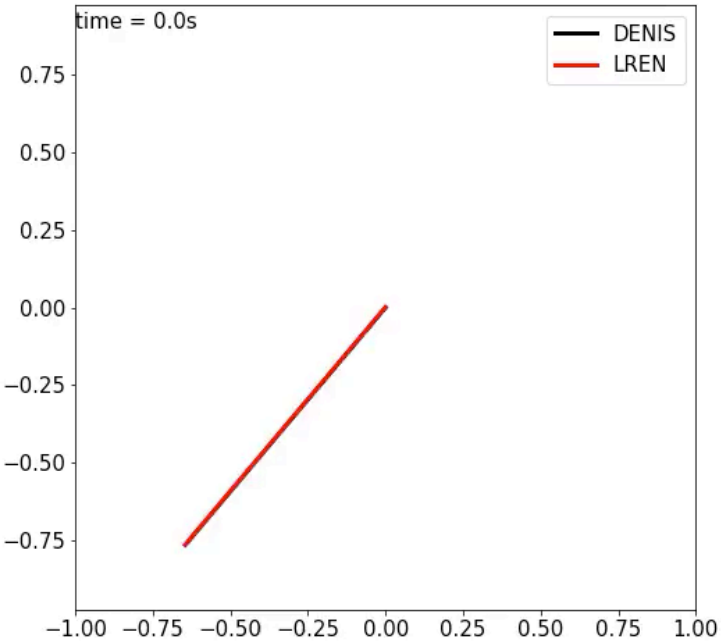
DENIS VS LREN Showdown

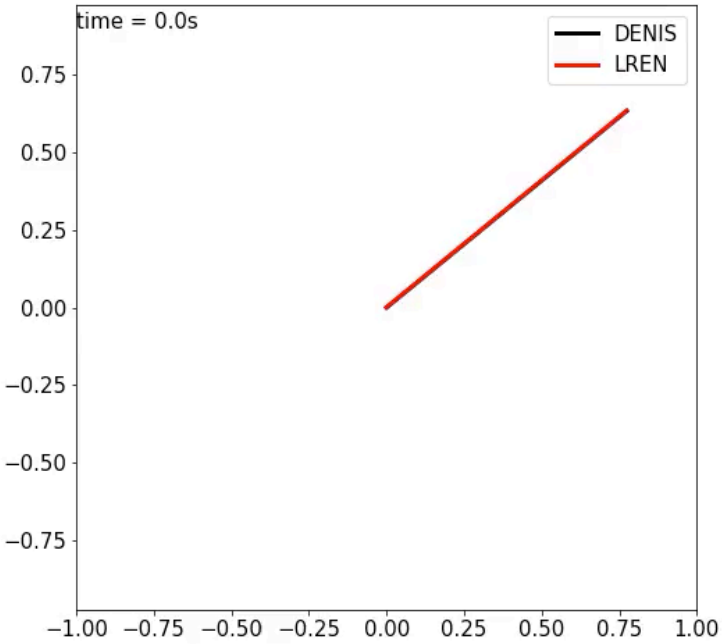
How about KOOC? - Same as before, LQR on the latent dims, actual system dynamics



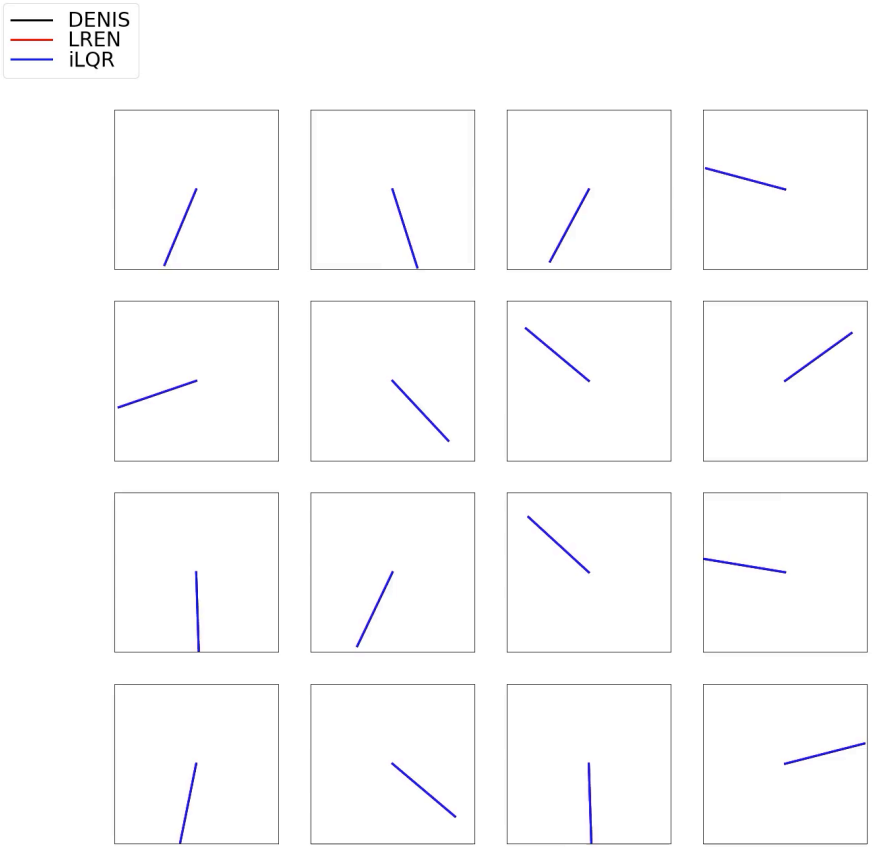
DENIS VS LREN Showdown

How about KOOC? - Same as before, LQR on the latent dims, actual system dynamics



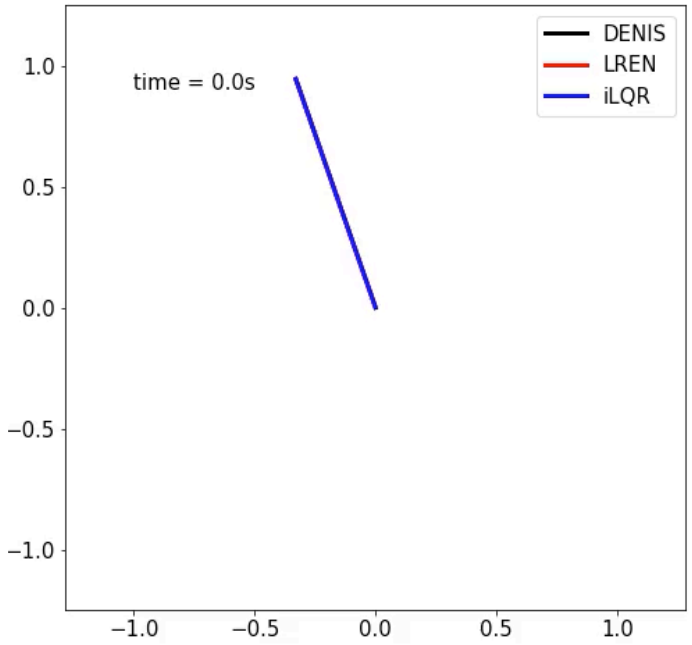


DENIS VS LREN Showdown



erative LQR (iLQR)?

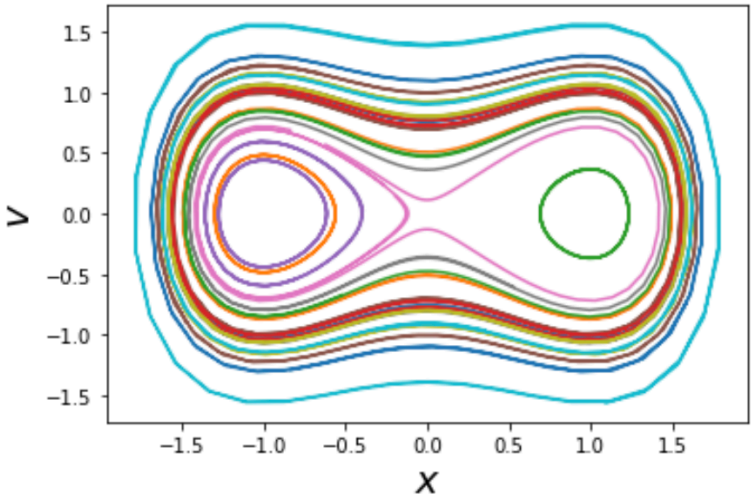
Example: LREN blows up under control input

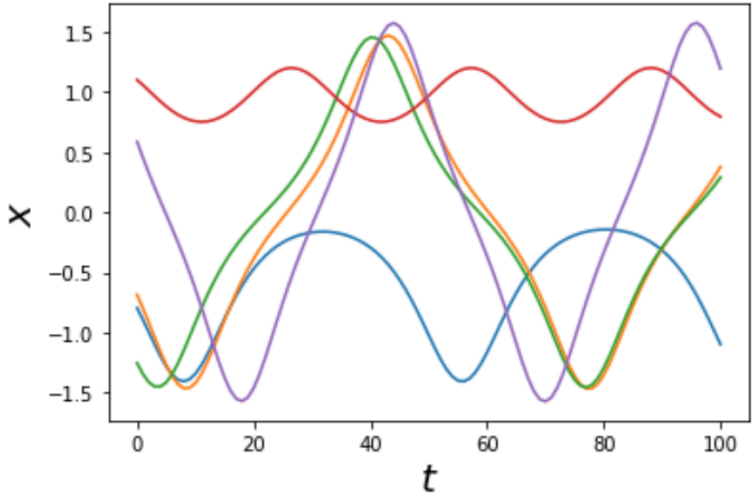


Next Step: Duffing Oscillator & New Architecture?

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ x_1 - x_1^3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$\mathcal{H} = \frac{1}{2}x_2^2 - \frac{1}{2}x_1^2 + \frac{1}{4}x_1^4$$



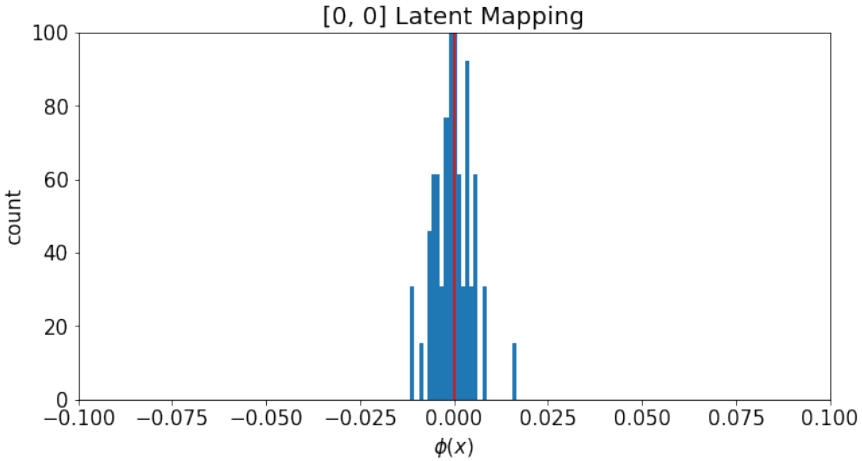


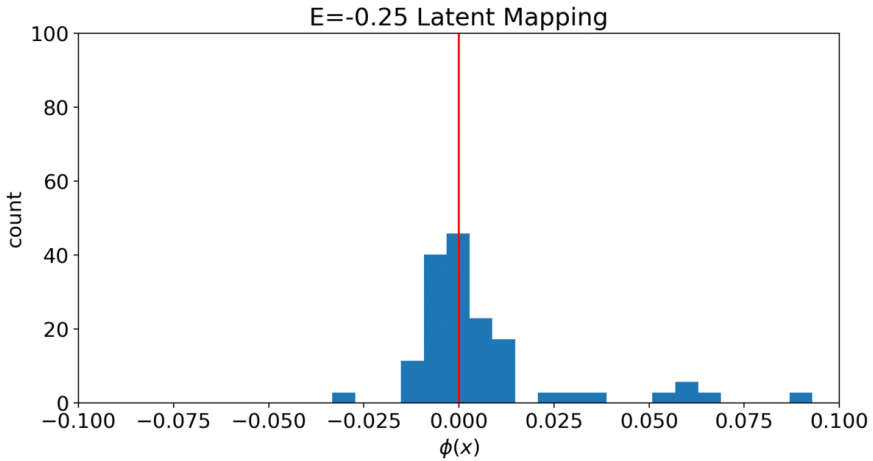
UNIVERSITY OF CAMBRIDGE

Project Summary W3-4

1. Does Phi(0)=0?

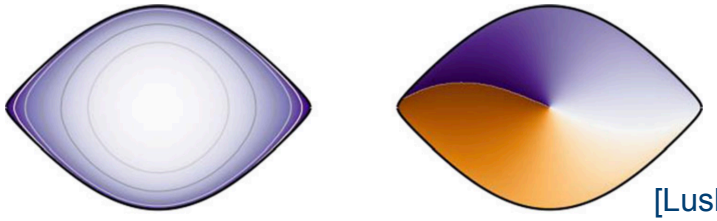
- Answer: Not really.





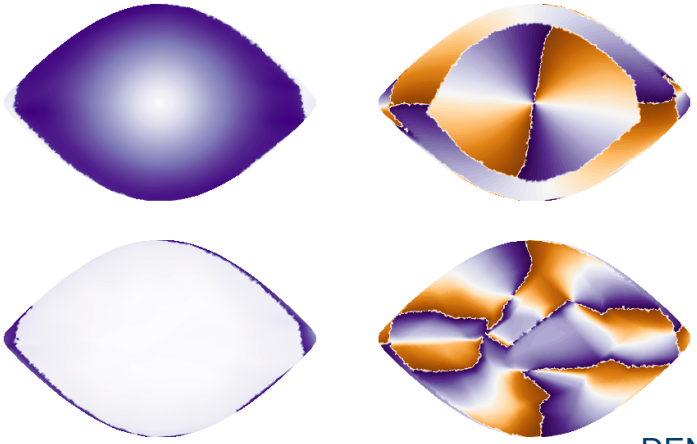
Discontinuity Issues

- Not easy to solve :(

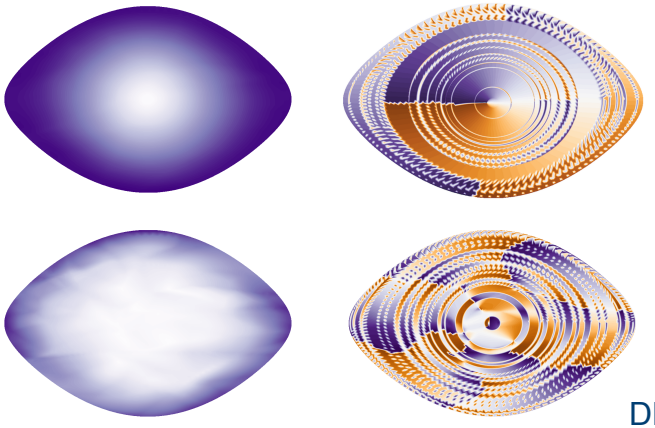


[Lush et al.]

$$E = m\frac{\left(l\dot{\theta}\right)^2}{2} + mgl(1-\cos(\theta))$$



DENIS



DENIS, smoothed

MPC?

- Done, but bugs? Parameters? Still not there yet….

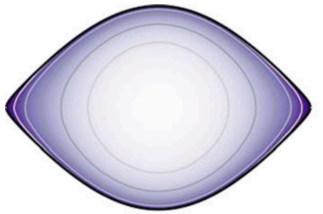
$$\begin{aligned} &\underset{U\in\mathbb{R}^{mN_p}}{\text{minimize}} && U^\top H U^\top + h^\top U + z_0^\top G U \\ &\text{subject to} && LU + M z_0 \leq c \\ &\text{parameter} && z_0 = \psi(x_k) \end{aligned}$$

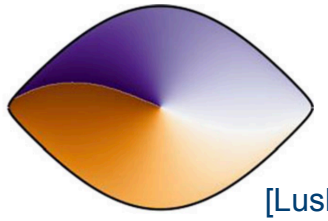
UNIVERSITY OF CAMBRIDGE

Project Summary W4-5

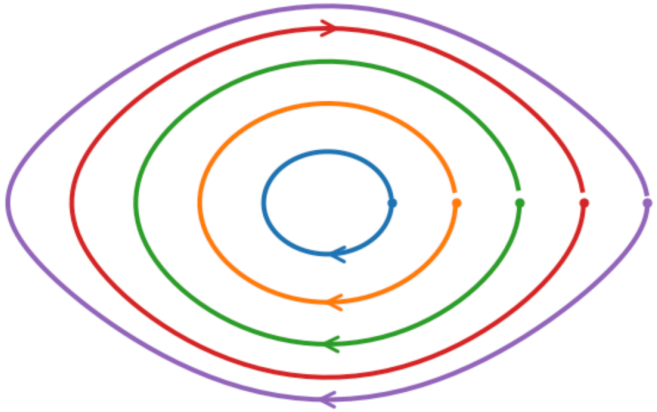
1. Discontinuity Issues

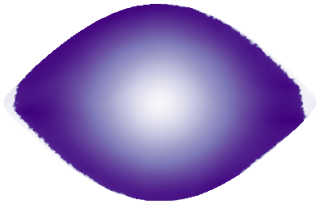
- Behold…

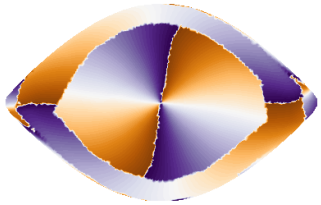


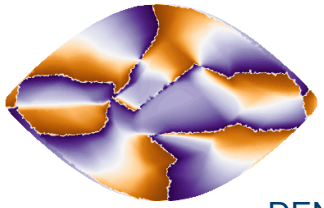


[Lush et al.]

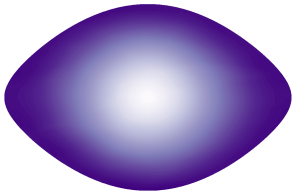


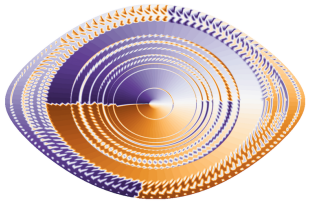


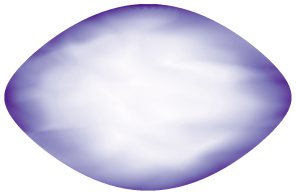


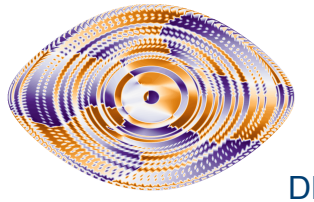


DENIS

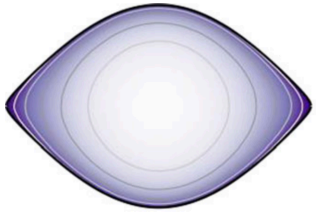


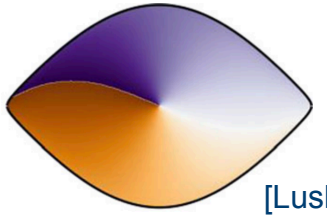




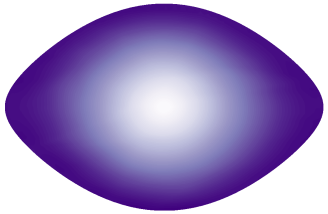


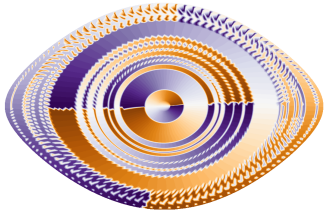
DENIS, smoothed



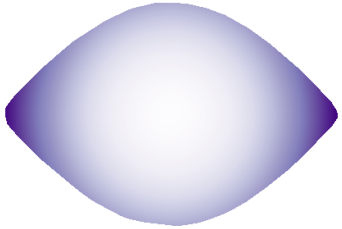


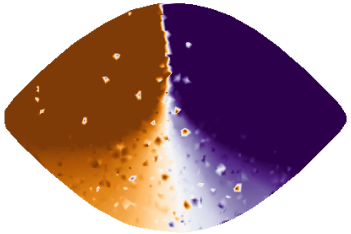
[Lush et al.]





DENIS, previous

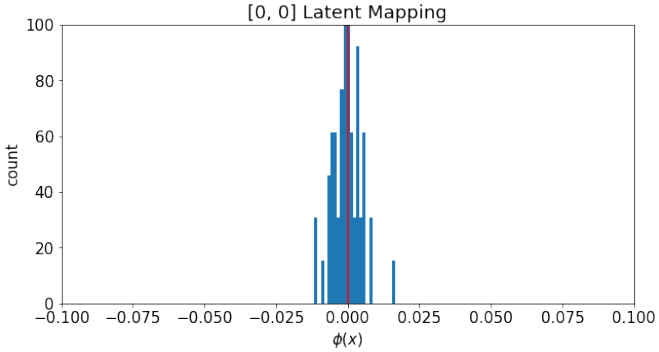


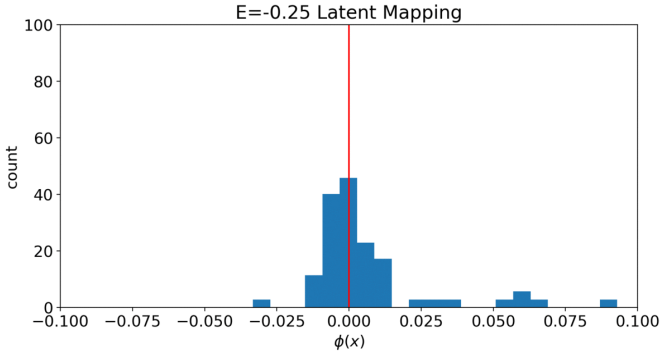


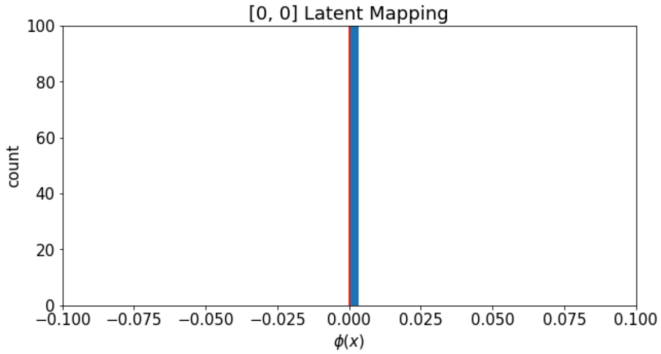
DENIS, Now

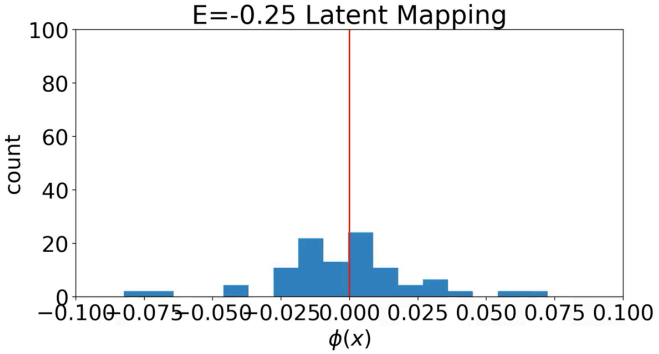
2. Does Phi(0)=0?

- Answer: Yes $\mathscr{L}_{\text{new}} = \mathscr{L}_{\text{old}} + \alpha \frac{1}{N} \sum_{n}^{N} \|K_0^n \phi(\mathbf{0})\|_2$



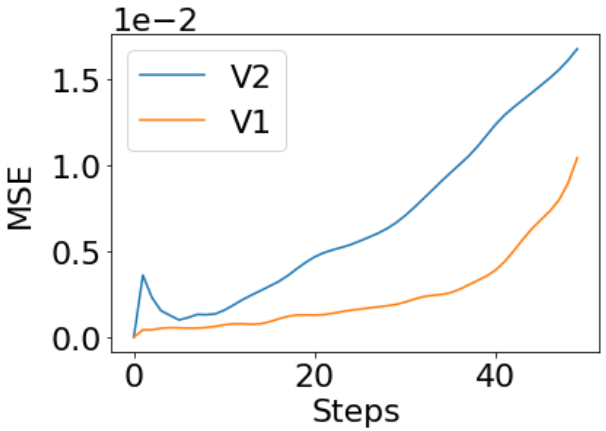


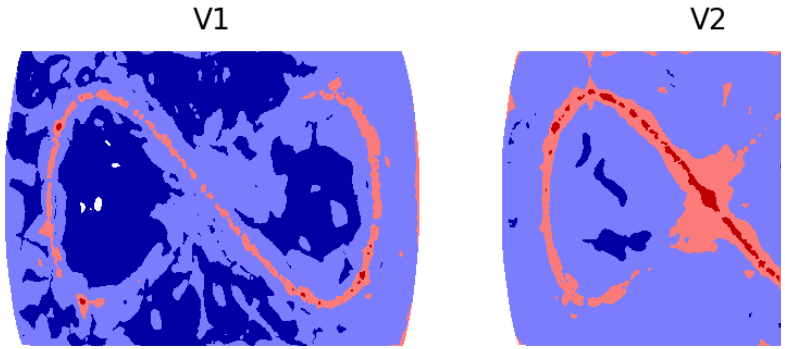




3. Now Phi(0)=0, is MPC/LQR Better?

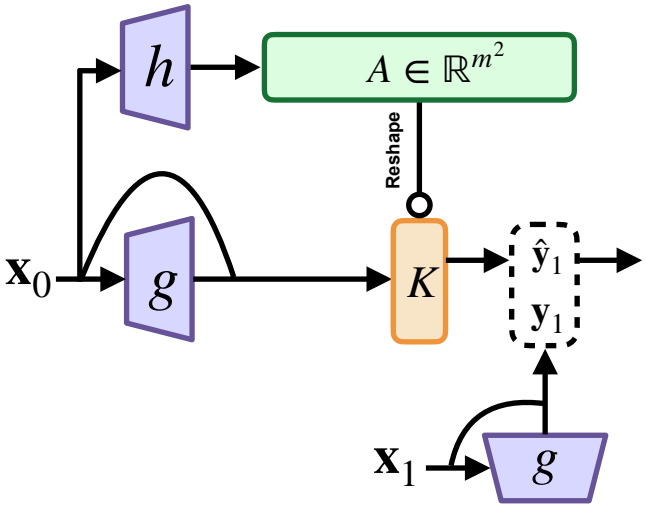
- Not really…





3. Curse of Dimensionality And Discontinuity

- If $\phi(x) \in \mathbb{R}^m$ then the auxiliary network has $\mathcal{O}(m^2)$ parameters….
- Sat $x \in \mathbb{R}^N$ and let $m = 50N$, network will now have $\mathcal{O}(2500N^2)$ parameters :-o
- Current network has 120K parameters just for a 2D problem
- Discontinuity as both h and g may work together to map eigenfunctions to different locations
- Is there a way to 1. Reduce number of parameters 2. Learn the eigenfunctions



- Jacobian form is not possible - we include our state :(
- Is there a way to globally align the eigenvectors?
- Solutions:

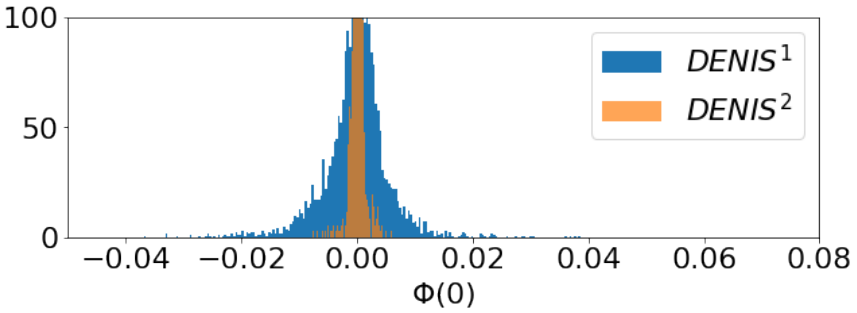
UNIVERSITY OF CAMBRIDGE

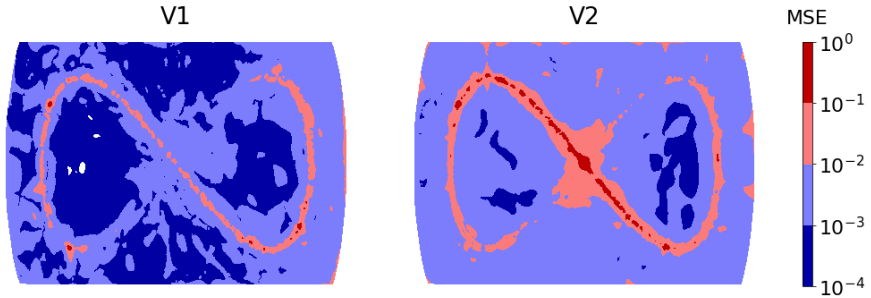
Project Summary W4-5

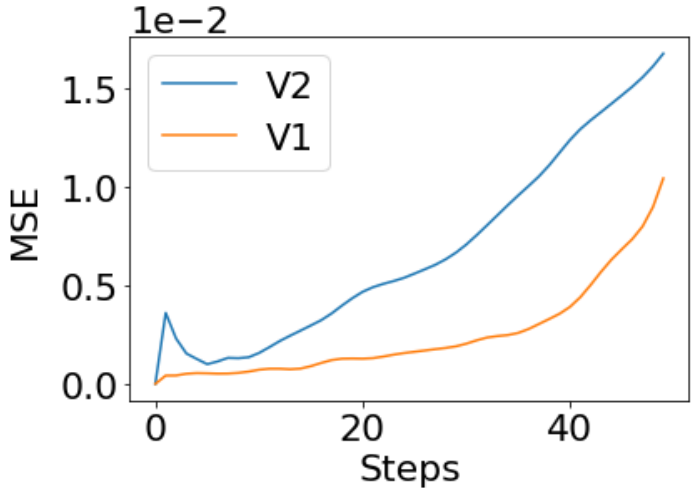
Wrapping up: Denis's Journey - V2

V1 VS V2: From last session: what is the benefit of adding the loss term?

$$\mathscr{L}_{\text{new}} = \mathscr{L}_{\text{old}} + \alpha \frac{1}{N} \sum_{n}^{N} \|K_0^n \phi(\mathbf{0})\|_2$$

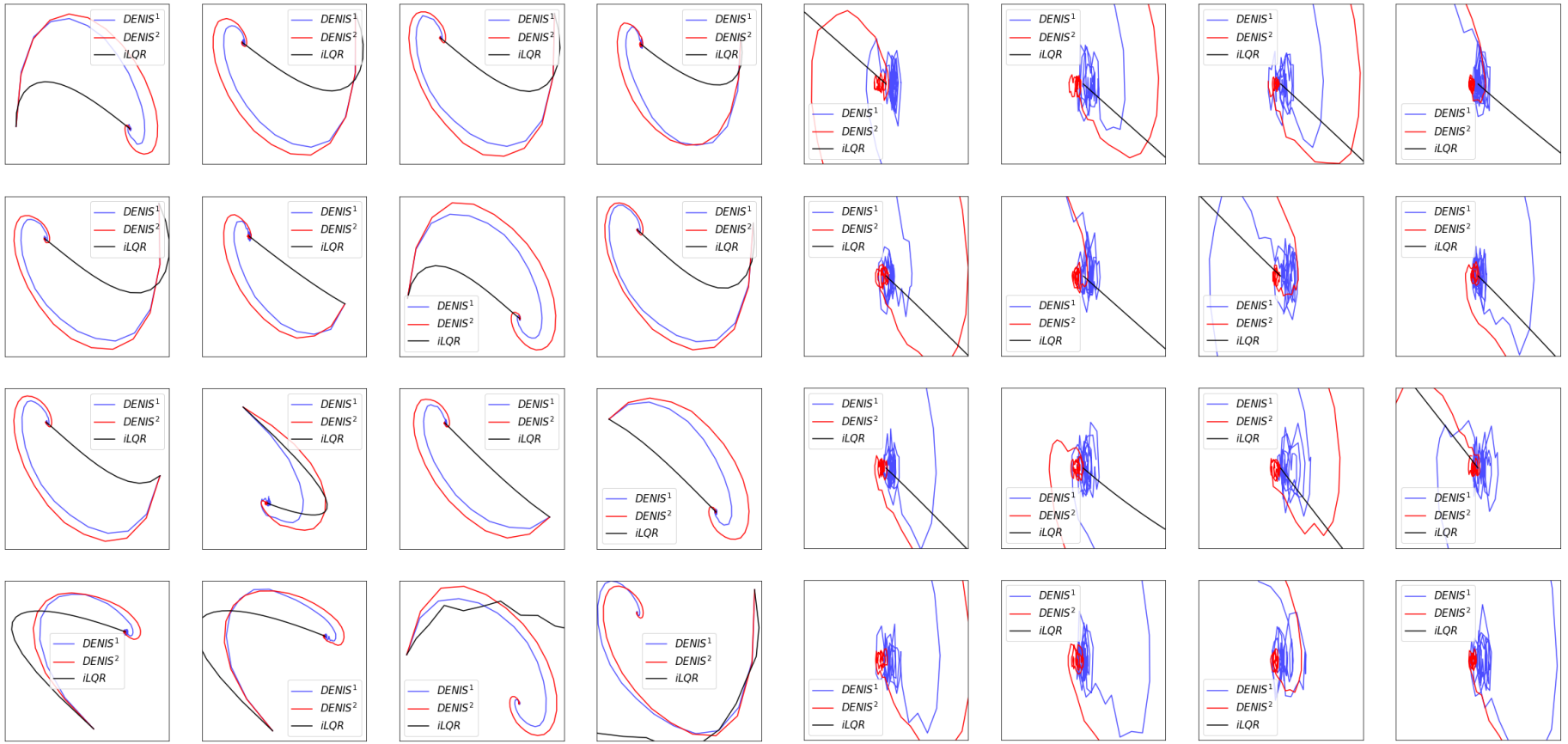






Wrapping up: Denis's Journey

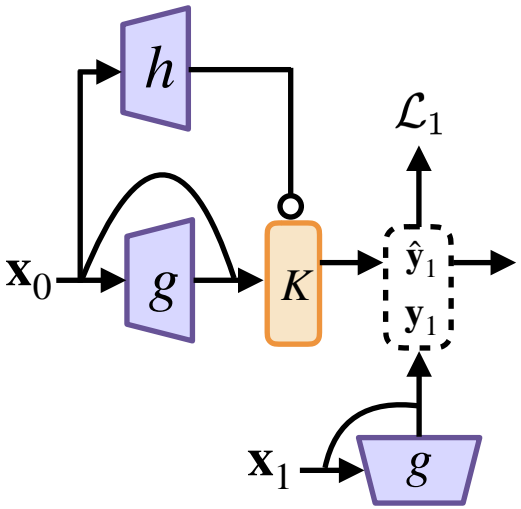
V1 VS V2: From last session: what is the benefit of adding the loss term?

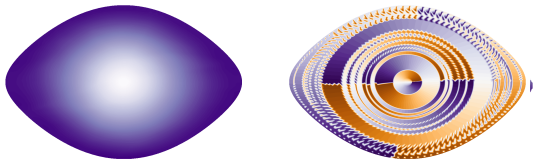


Wrapping up: Denis's Journey -V3

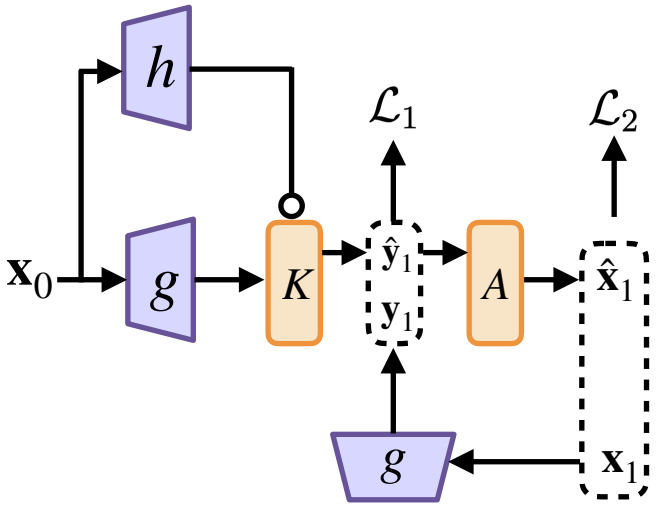
Motivation: Failed to smooth plots, is there a way to 'guide/force' the network towards learning the energy function while keeping continuity?

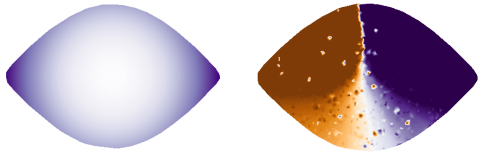
V1 / V2



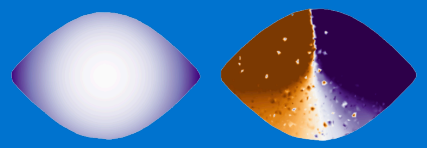


V3





Wrapping up: Denis's Journey



V3: Last attempt on discovering the energy function

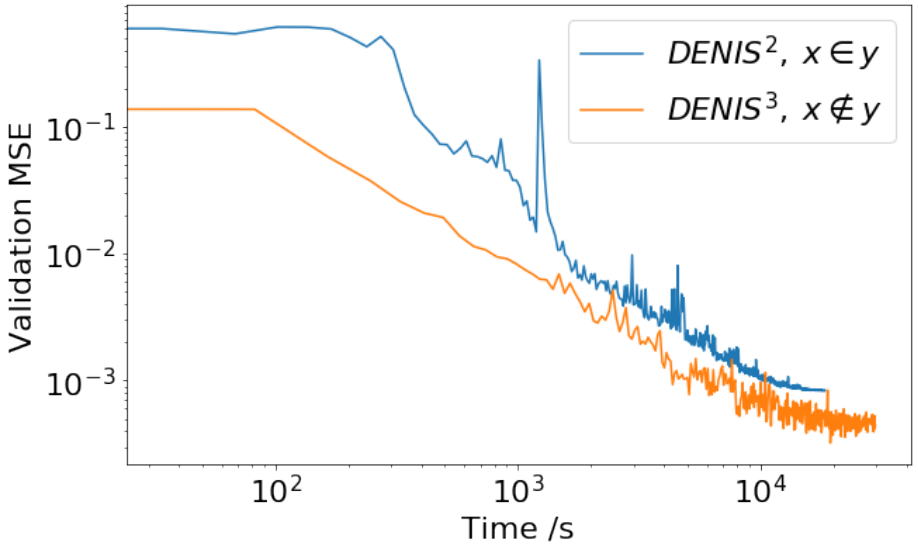
$$K = \begin{bmatrix} B_0 & & & \\ & B_1 & & \\ & & B_2 & \\ & & & \ddots \\ & & & & B_N \end{bmatrix}$$

$$B_N = \exp(\mu_n \delta t) \begin{bmatrix} \cos(\omega_n \delta t) & -\sin(\omega_n \delta t) \\ \sin(\omega_n \delta t) & \cos(\omega_n \delta t) \end{bmatrix}$$

$$\omega_n = 0 \; \forall n \in \{1, \cdots N\}$$

What is its performance and How is it to train?

Both approximately same number of parameters (~100K),
Both latent dimension is 150



Wrapping up: Denis's Journey

V3: How about MPC?

$$x^+ = f(x) + Bu$$

$$y^+ = Ky + \tilde{B}u$$

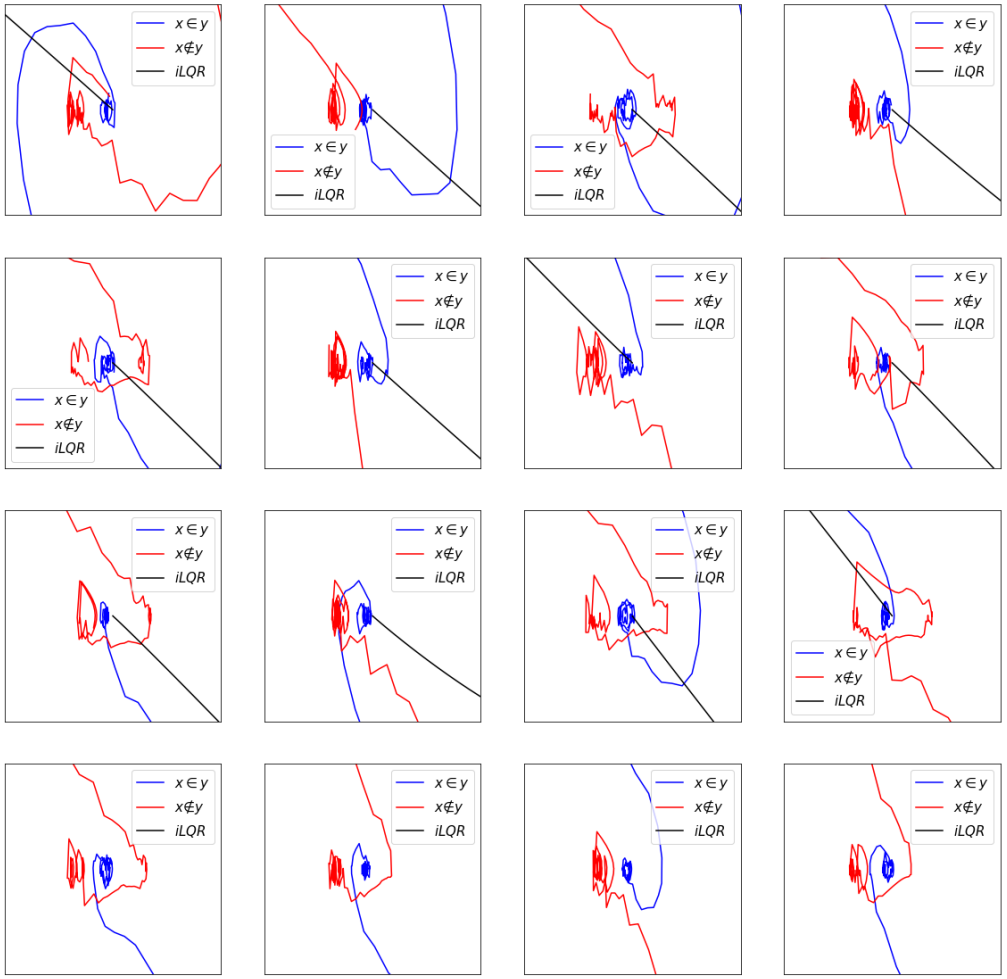
$$y_0 = \Phi(x_0)$$

$$\hat{x} = Ay$$

Easy to show:

$$\tilde{B} = A^{\dagger}B$$

$$\tilde{Q} = A^T Q A$$



0.1

What have we learnt?

- $\Phi(0) = 0$ is ideal if we want to drive system to 0 and stay there, if this is violated, we may get noisy behaviour at 0. However, we may need to trade off model performance
- $x \in y$ is slightly worse than $x \notin y$ in terms training performance, control wise is uncertain
- Reconstructing the energy function is difficult without a non-linear decoder, but does it really matter?
- The idea behind DENIS is especially useful for MPC
- MPC is pretty good

Introducing: DEINA

Double **E**ncoder for **I**nput **N**on-**A**ffine Systems

Theory reminder:

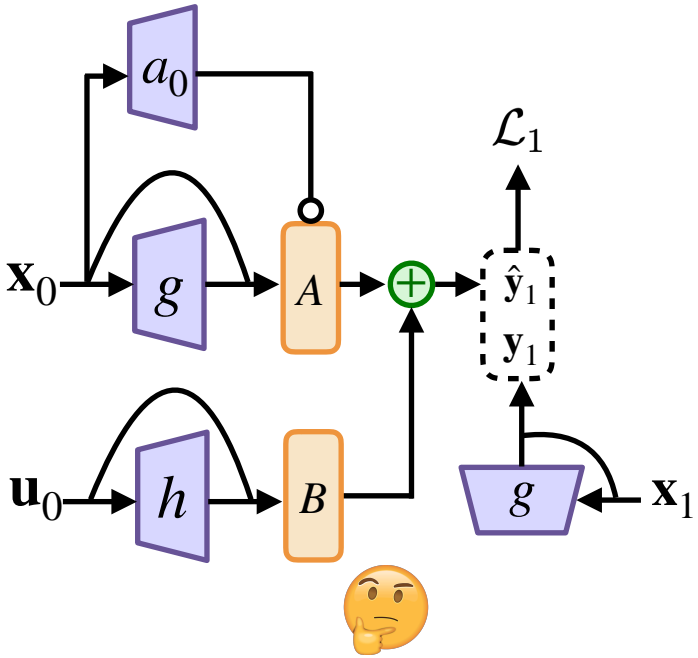
$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$$

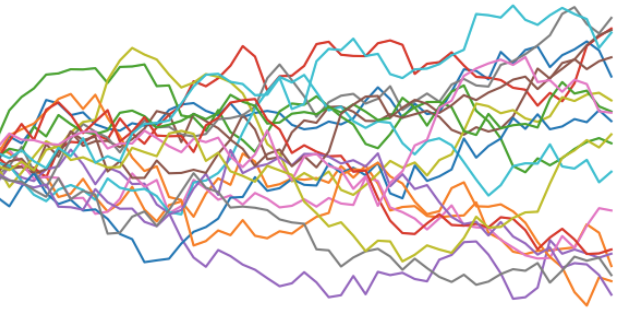
$$\mathbf{y}_0 = \Phi_x(\mathbf{x}_0)$$

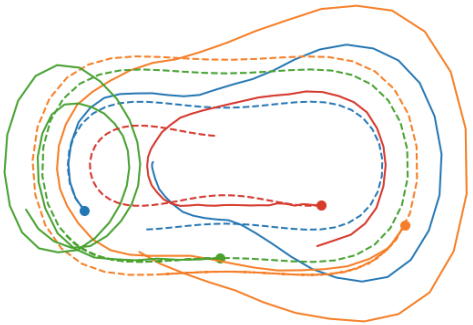
$$\mathbf{q} = \Phi_u(\mathbf{u})$$

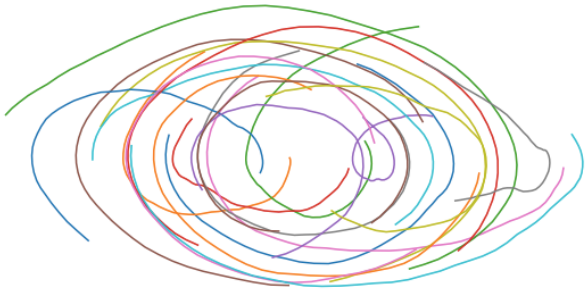
$$\mathbf{y}_{k+1} = A\mathbf{y}_k + B[\mathbf{u} \;\; \mathbf{q}]^T$$

[Mezic]





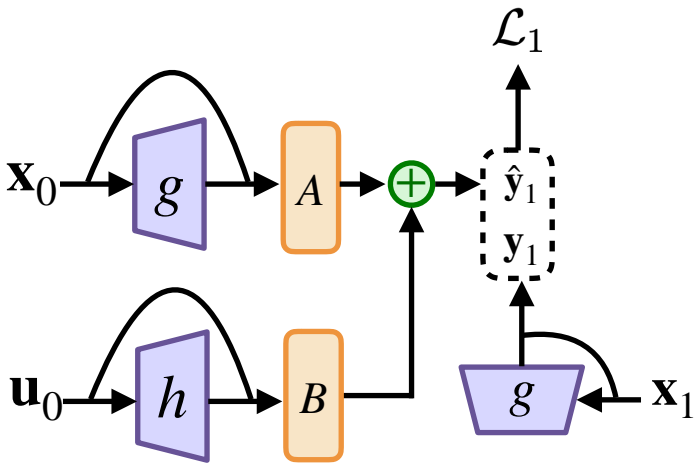


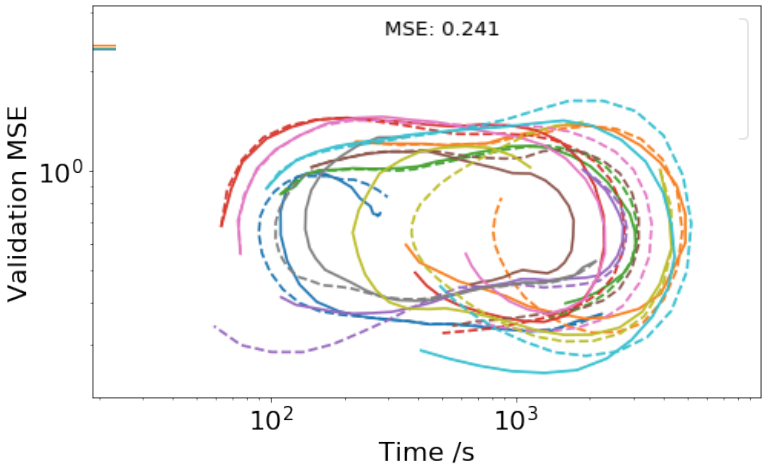


Initial Results - TL;DR Not Good

Questions:

1. Can we train only with dynamics with input, will the network generalise to 0 input dynamics?
2. Do we parameterise? If so, what information do we give to the auxiliary?
3. What type of input noise do we provide? (Brownian/ Gaussian/ Uniform, etc?)





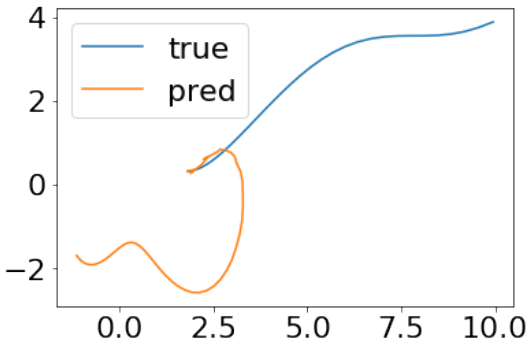
Input Affine: Trivial Extension from DENIS?

Not so much….

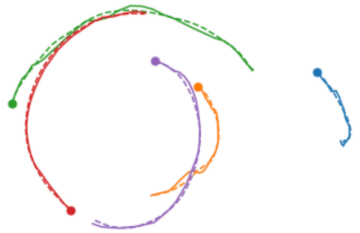
$$\mathbf{y}^1 = A\mathbf{y}^0 + B\mathbf{u}^0$$

$$\mathbf{y}^N = A^N\mathbf{y}^0 + \sum_{i=0}^{N-1} A^i B u^{i(N-1)}$$

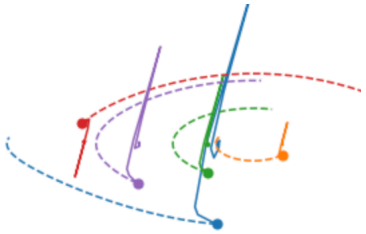
DENIS Post training modification



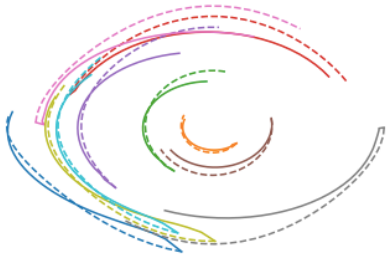
DENIS+, w Brownian inputs

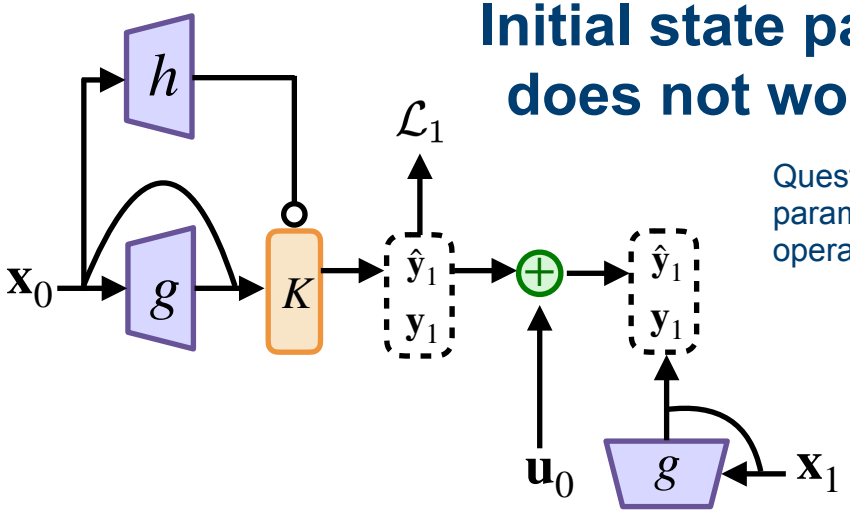


Trained DENIS+ on 0 inputs



DENIS+, w Brownian + 0 inputs





Initial state parameterisation does not work with inputs!

Question: Did we learn a method to parameterise the Koopman operator for brownian motion?

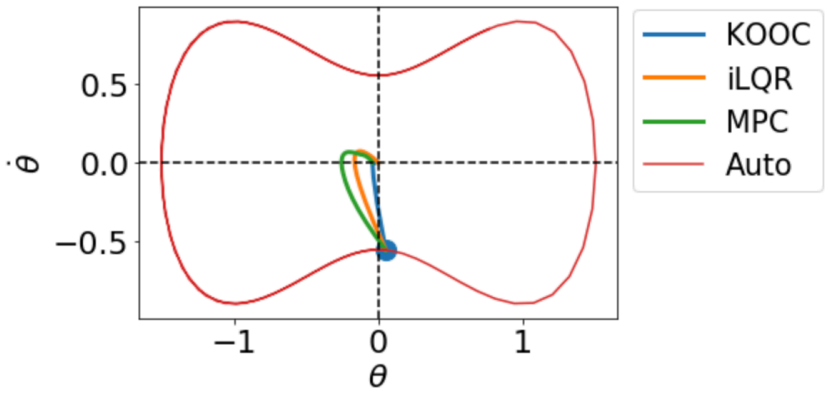
UNIVERSITY OF CAMBRIDGE

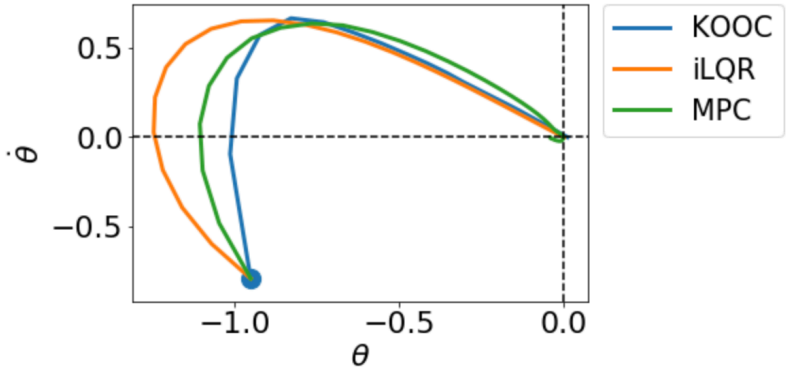
Project Summary W8-9

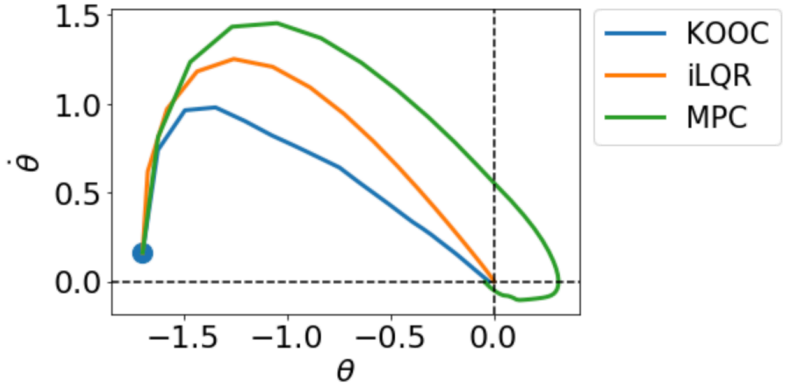
DENIS

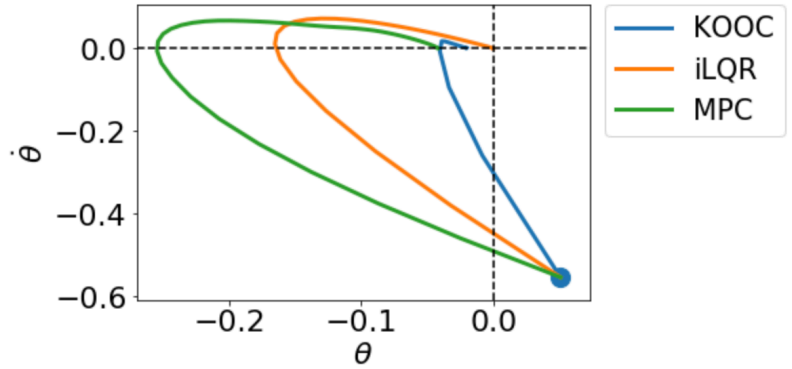
Once Upon a Time…

- Bug discovered with DENIS KOOC…. Now KOOC beats iLQR (I have been using the same koopman operator for every initial conditions!)





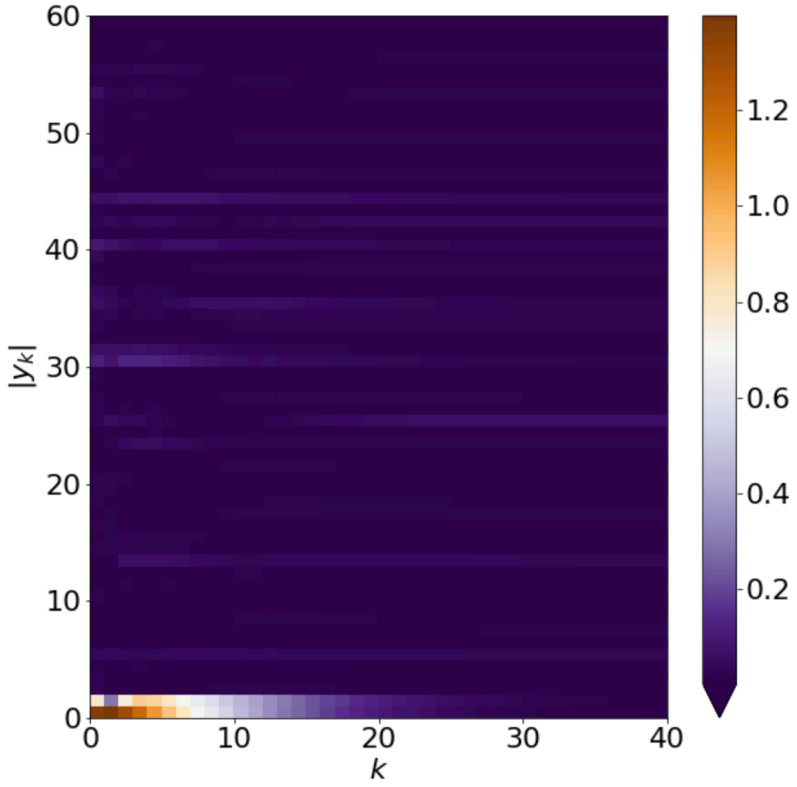


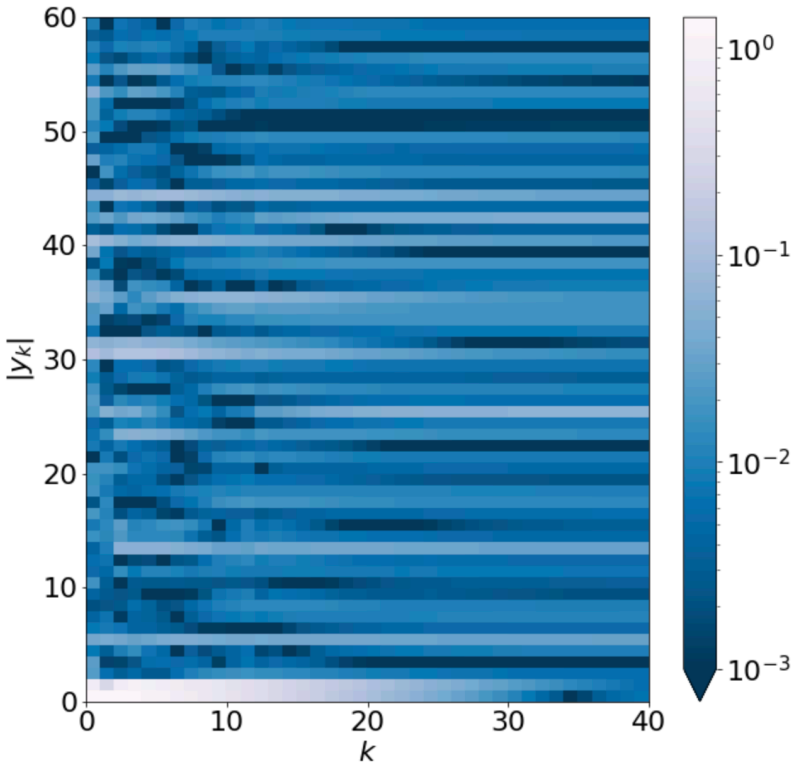


DENIS

Once Upon a Time…

- How about the latent space?

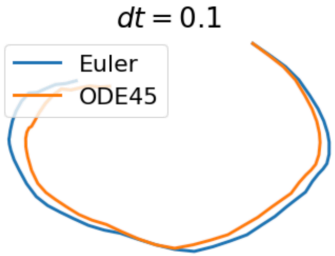


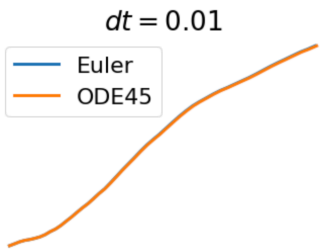


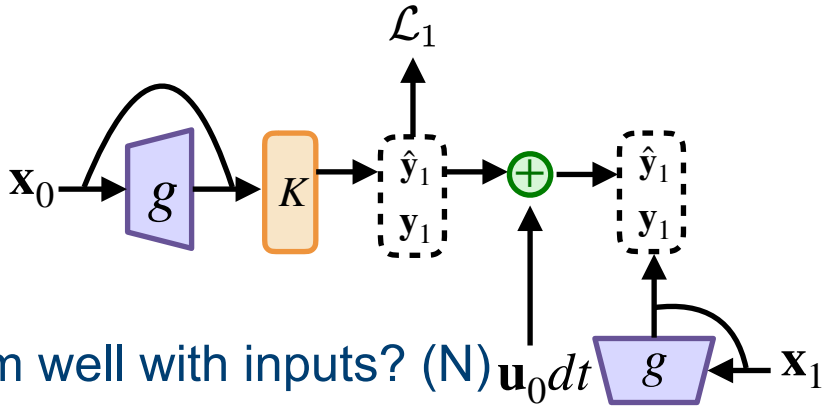
LREN

Sanity check for input affine systems

- We have shown, at large dt, LREN cannot be modified to include inputs due to instability. At small dt, does it perform well? (Y)







- Does LREN trained without inputs perform well with inputs? (N)

$$\mathbf{y}^N = A^N \mathbf{y}^0 + \sum_{i=0}^{N-1} A^i B u^{i(N-1)}$$

- Does LREN trained with inputs perform well without inputs? (Y)

LREN

Data Generation

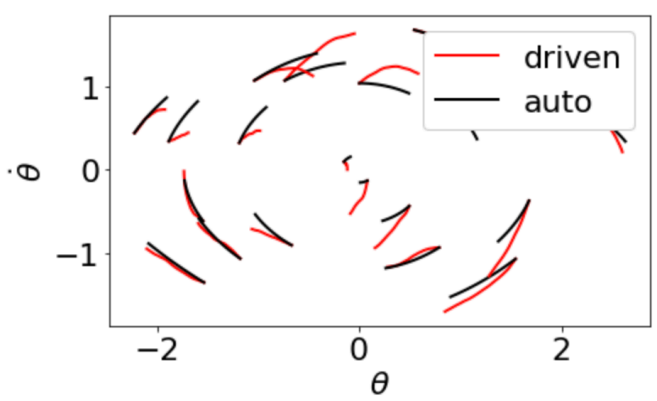
- Some time dedicated to write a general ‘data generator’ method (a necessary evil)

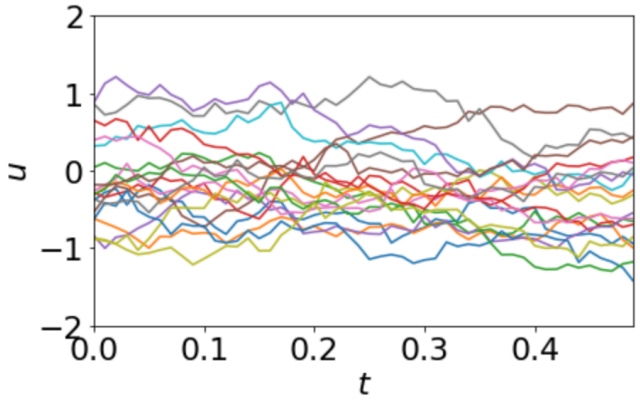
```
N = 25000
n_steps = 50
dt = 0.01

generator = Data_Generator(pendulum_dynamics, dt, 2)
```

- Inputs are now drawn from OU processes $X_{n+1} = X_n + \theta\left(\mu - X_n\right)\Delta t + \sigma \Delta W_n$

$$\mu \sim U(-1, 1), \quad \sigma = 1, \quad \theta = 2$$





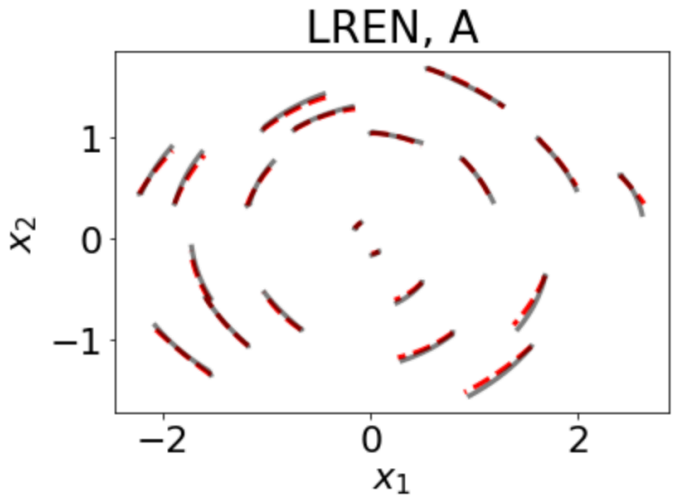
LREN

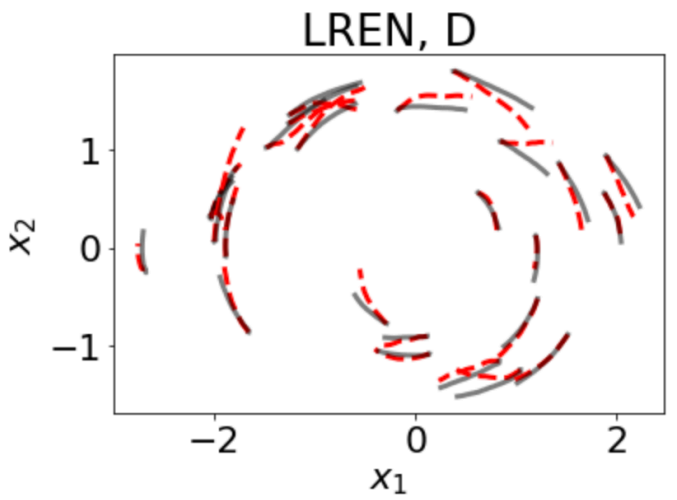
Results

A: Autonomous trajectories, no input
D: Driven trajectories, with input

LREN: Base LREN model
LREN + : LREN model, with added input during training, trained with balanced A and D trajectories
LREN - : Same as previous, but only trained on D trajectories

- Does LREN trained without inputs perform well with inputs?



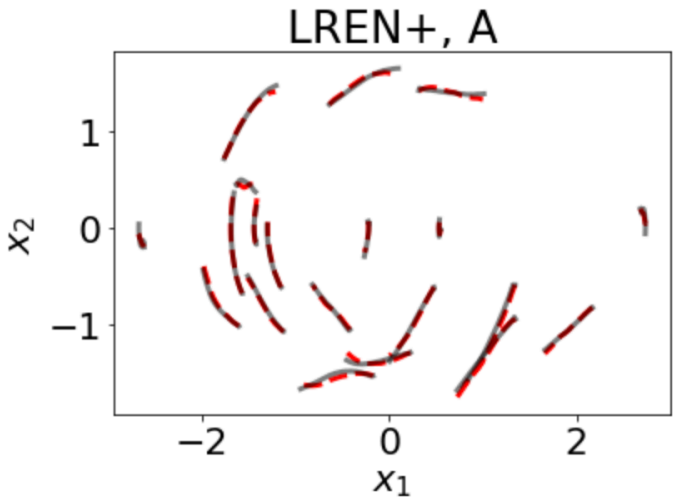


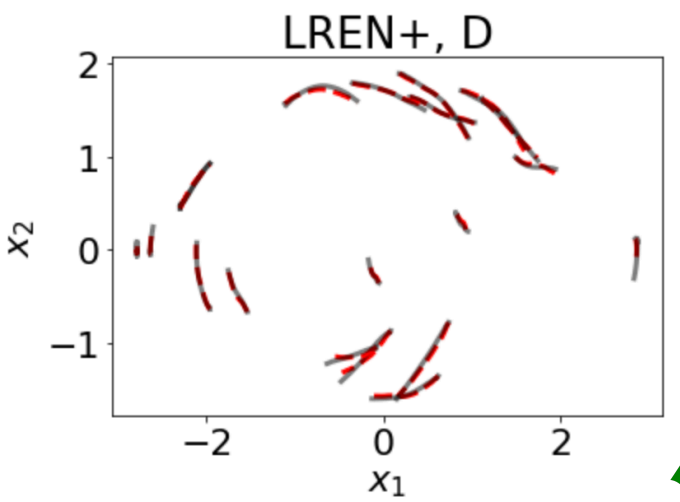
LREN
Results

A: Autonomous trajectories, no input
D: Driven trajectories, with input

LREN: Base LREN model
LREN + : LREN model, with added input during training, trained with balanced A and D trajectories
LREN - : Same as previous, but only trained on D trajectories

- What if we allow LREN model to train with inputs? (A+D Dataset)





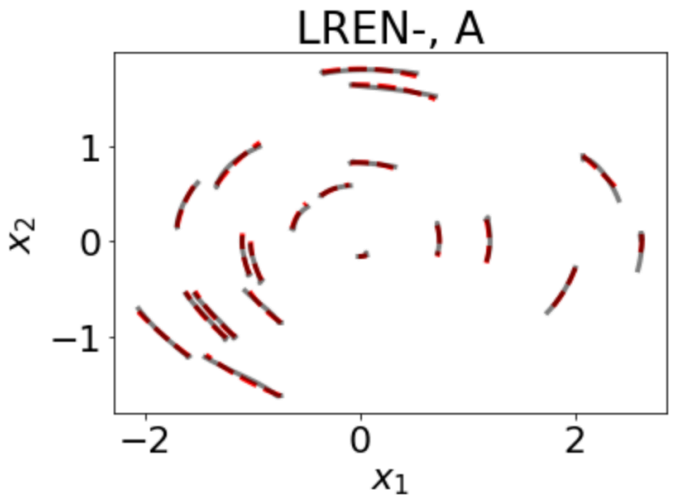
LREN

Results

A: Autonomous trajectories, no input
D: Driven trajectories, with input

LREN: Base LREN model
LREN + : LREN model, with added input during training, trained with balanced A and D trajectories
LREN - : Same as previous, but only trained on D trajectories

- What if the model is trained only on driven data? Can it reconstruct autonomous trajectories?



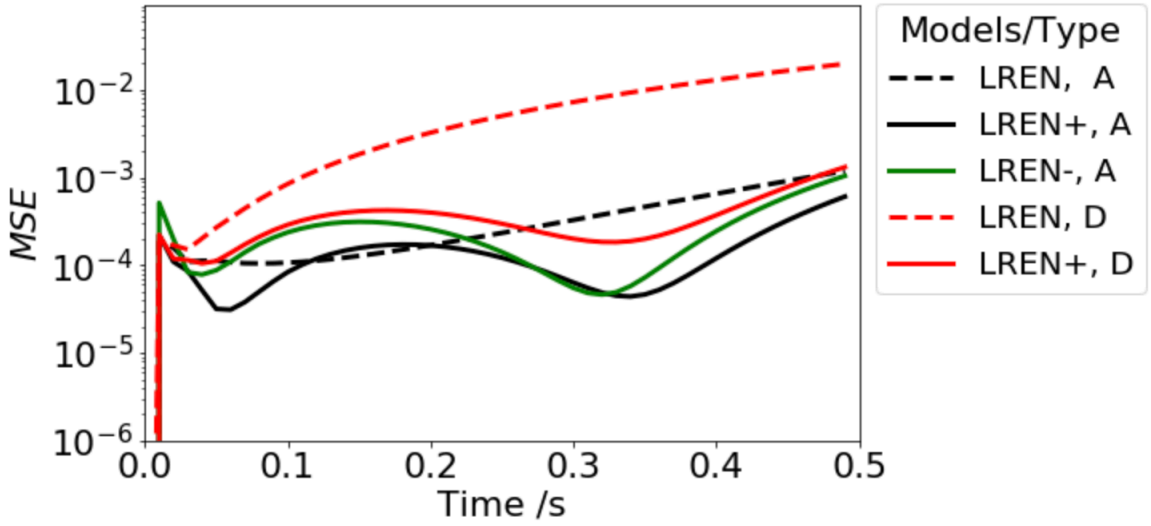
LREN

Results

A: Autonomous trajectories, no input
D: Driven trajectories, with input

LREN: Base LREN model
LREN + : LREN model, with added input during training, trained with balanced A and D trajectories
LREN - : Same as previous, but only trained on D trajectories

- What about overall performance across all methods investigated?

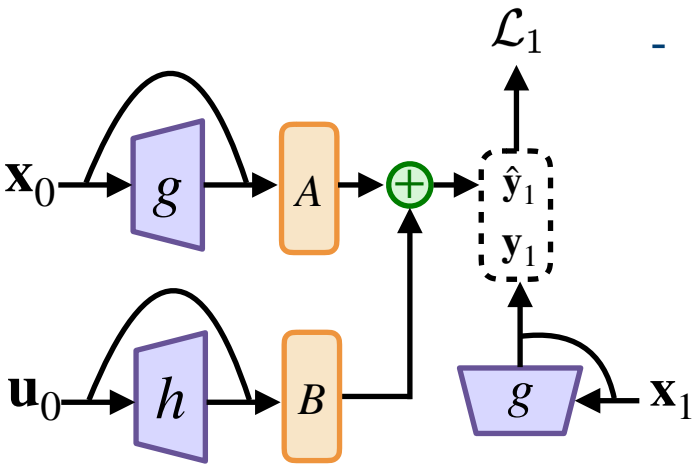


- At SMALL time steps:
- Training without inputs will not predict driven trajectories
- Training with inputs may even improve autonomous performance when dataset is balanced
- Training with only driven trajectories may slightly hurt autonomous prediction

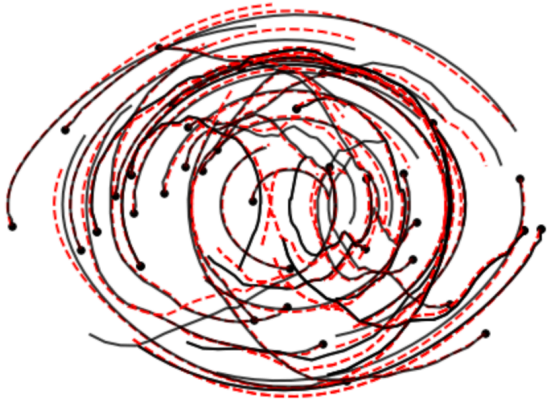
DEINA

Double **E**ncoder for **I**nput **N**on-**A**ffine Systems

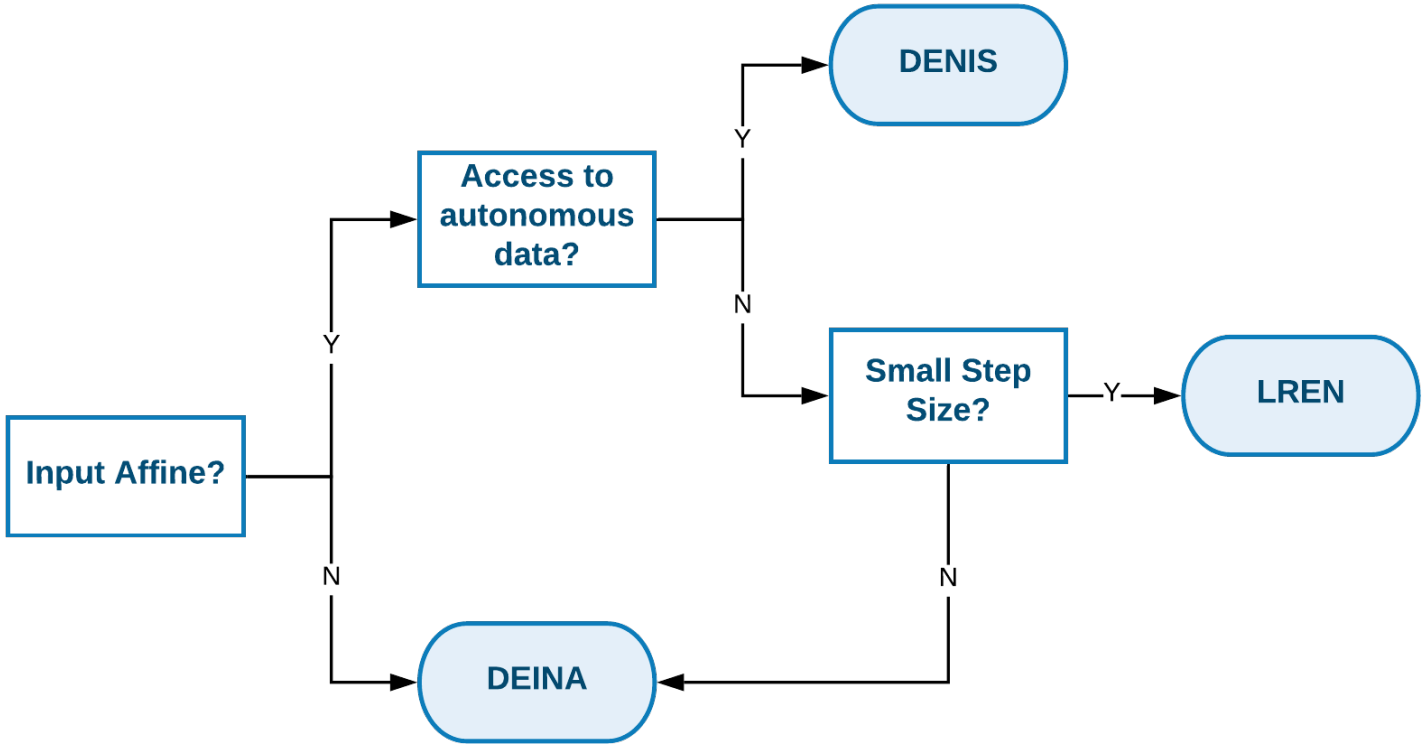
- Bug fix, architecture is confirmed



- Why do we need to append input?
  - Without it, we will need to include a decoder (MPC, LQR will not work as we do not know how to reconstruct the latent inputs)
  - Because of the 'many to one' issue (Rory), the decoder must be linear
  - A linear decoder is equivalent to appending input



DL Koopman Decision Tree:



UNIVERSITY OF CAMBRIDGE

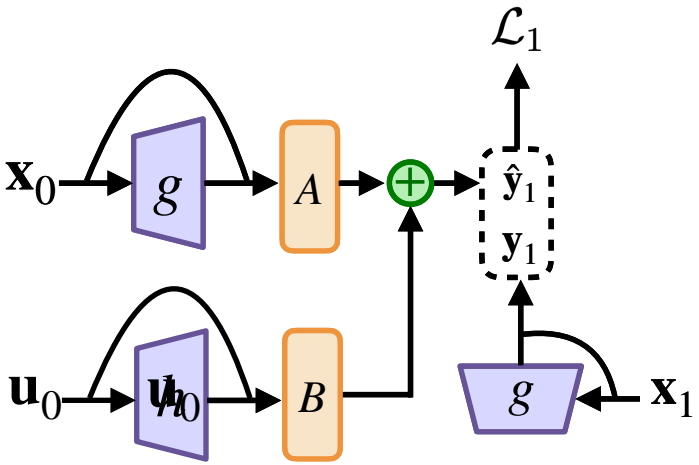
Slides Remaining: 7

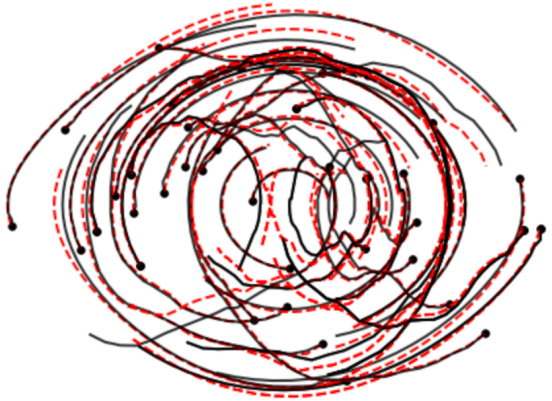
Project Summary Week ∞

DEINA DEINA DEINA

DEINA: First Iteration

- $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{u} \in \mathbb{R}^M$, $A \in \mathbb{R}^{\hat{N}\times\hat{N}}$, …
- B operates on $[\mathbf{u}, h(\mathbf{u})]^T$ but is optimal control obvious/sensible?
- i.e. For the LQR gain matrix $G \in \mathbb{R}^{\hat{N}\times\hat{N}}$, can we simply ditch the columns to get $G \in \mathbb{R}^{\hat{N}\times M}$?
- Or…. Shall we just ditch h all together?



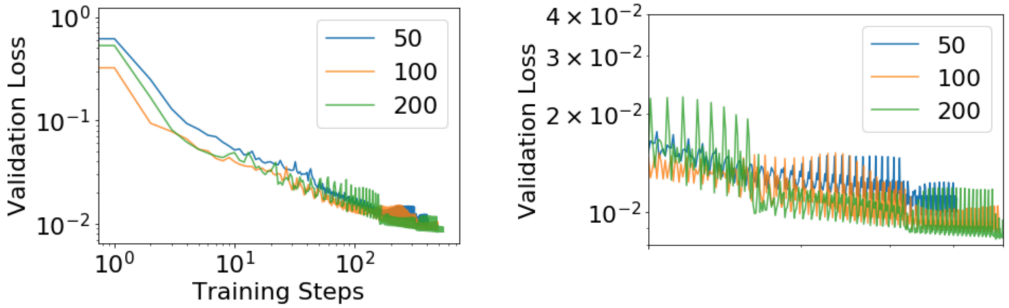


RMSE: 0.094 c.f. 0.024, 30TH@0.1s

DEINA: First Iteration

- So…. Results not great
- Increase model capacity? Fix encoder dimension to be $2 \times \frac{N}{4} \times \frac{N}{2} \times \frac{N}{2} \times N$
- LR Scheduler: lr*=0.5 per 20 epochs

N	Epochs	RMSE
50	50	0.090
100	40	0.089
200	80	0.085
300	100	0.085



Work's focus: make DEINA better

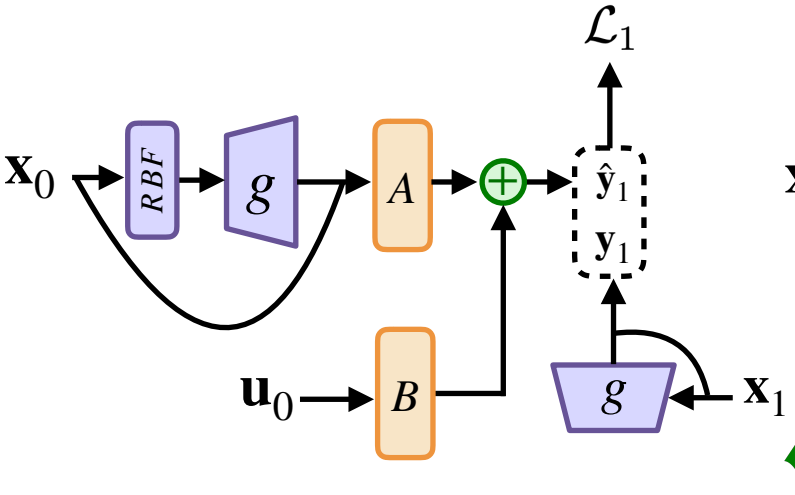
- It appears size doesn't matter?? How?

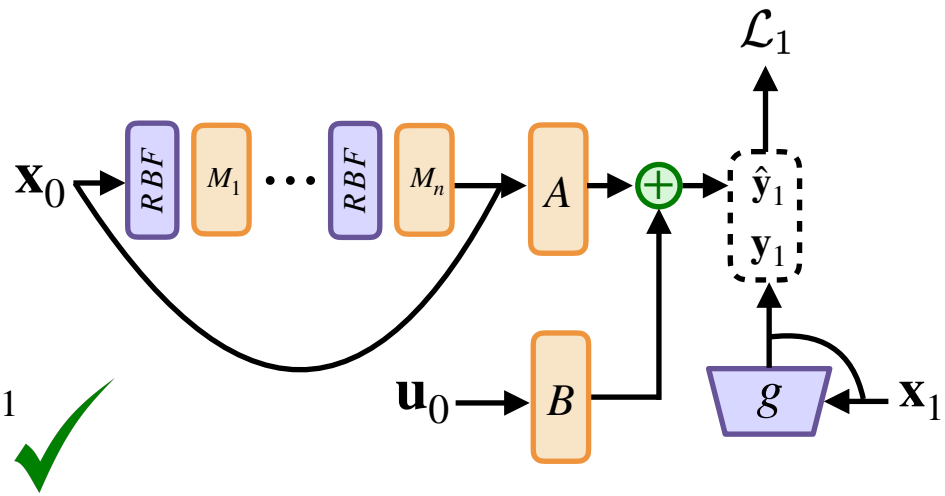
DEINA RBF

- Is there another way to increase the capacity of the network other than upping the latent?
- Make the model more “non-parametric”
- RBF Networks?

$$\begin{bmatrix} \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_n \end{bmatrix} = \begin{bmatrix} \phi\left(\sigma_1 \parallel \boldsymbol{x}_1 - \boldsymbol{c}_1 \parallel_2\right) & \cdots & \phi\left(\sigma_m \parallel \boldsymbol{x}_1 - \boldsymbol{c}_m \parallel_2\right) \\ & \vdots & \\ \phi\left(\sigma_1 \parallel \boldsymbol{x}_n - \boldsymbol{c}_1 \parallel_2\right) & \cdots & \phi\left(\sigma_m \parallel \boldsymbol{x}_n - \boldsymbol{c}_m \parallel_2\right) \end{bmatrix}$$

N	Epochs	RMSE
50	50	0.090
100	40	0.089
200	80	0.085
300	100	0.085
RBF - 100	50	**0.071**





DEINA ODE

- Something fun to do….
- Motivation: Maybe, we will get better result if the Koopman operator is continuous. Maybe, we can recreate DENIS behaviour (bounded long TH trajectories). No need for discrete LQR

Discrete Dynamics

$$\mathbf{y} = g(\mathbf{x}_{k+1})$$

$$\begin{bmatrix}\mathbf{x}_{k+1}\\ \mathbf{y}_{k+1}\end{bmatrix} = K\begin{bmatrix}\mathbf{x}_{k}\\ \mathbf{y}_{k+1}\end{bmatrix}$$

Continuous Dynamics

$$\dot{\mathbf{y}}_{t_1} = K\mathbf{y}_{t_1}$$

$$\mathbf{y}_{t_2} = \mathbf{y}_{t_1} + \int_{t_1}^{t_2} \dot{\mathbf{y}}_{t_1} dt$$

$$\mathbf{x}_{t_2} = C\mathbf{y}_{y_2}$$

Neural Ordinary Differential Equations

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bettencourt*, David Duvenaud
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

Backprop = Solving another ODE, but backwards in time!

$$\mathbf{a}(t) = \partial L/\partial \mathbf{y}(t) \qquad \mathbf{s}_0 = \left[\mathbf{y}\left(t_1\right), \frac{\partial L}{\partial \mathbf{y}\left(t_1\right)}, \mathbf{0}\right] \qquad \nabla = \begin{bmatrix} K\mathbf{y}(t) \\ -\mathbf{a}(t)^\top K \\ -\mathbf{a}(t)^\top \frac{\partial K}{\partial \theta} \end{bmatrix}$$

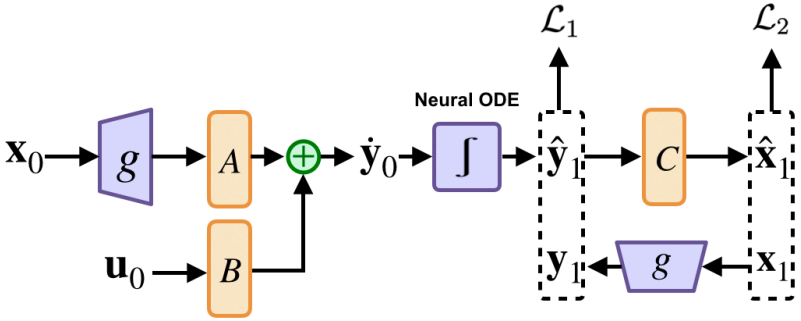
$$\left[\mathbf{y}\left(t_0\right), \frac{\partial L}{\partial \mathbf{y}\left(t_0\right)}, \frac{\partial L}{\partial \theta}\right] = \text{ODESolve}(\mathbf{s}_0, \nabla, t_1, t_0, \theta)$$

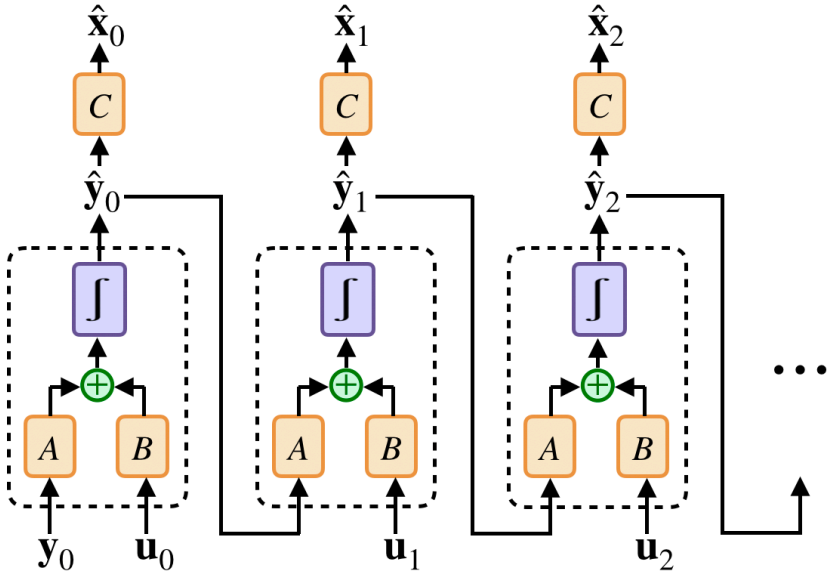
DEINA ODE

$$\dot{\mathbf{y}}_{t_1} = A\mathbf{y}_{t_1} + B\mathbf{u}_{t_1}$$

$$\mathbf{y}_{t_2} = \mathbf{y}_{t_1} + \int_{t_1}^{t_2} \dot{\mathbf{y}}_{t_1} dt$$

$$\mathbf{x}_{t_2} = C\mathbf{y}_{y_2}$$





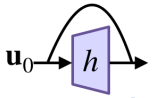
Initial Evaluation:

- (+) Converges within 2-5 epochs!
- (+) Smooth trajectories
- (-) RMSE does not improve, if not slightly worse than before :(

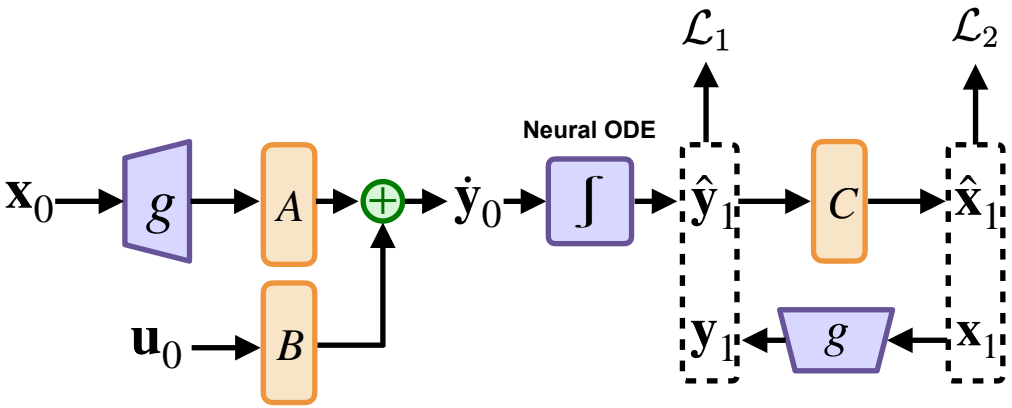
Summary & Future Work

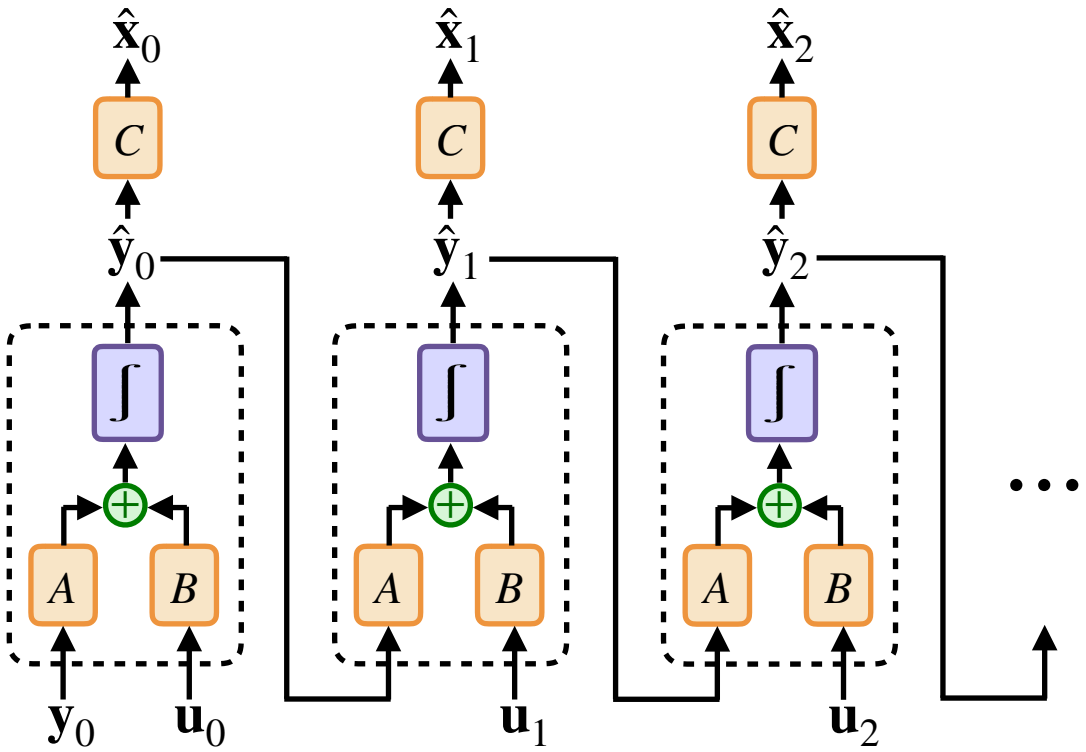
- DEINA, DEINA RBF, DEINA ODE presented
- DEINA RBF does increase model performance
- DEINA ODE is more like a proof of concept i,e. Continuous dynamics can be learnt using discrete data.
  - Are there more benefits to it?
  - Does it outperform previous methods with KOOC? Need to evaluate
  - Can MSE be improved with a RBF layer?

Summary & Future Work

- **Tidy up code, environment, write actual training script (for HP search), etc**
- Analyse these methods in more depth- MSE VS TH plots, etc
- Is KOOC with 

possible? Does it actually bring any benefit with model performance?
- Arm model, might be time?


$\mathcal{L}_1$
$\mathcal{L}_2$
Neural ODE
$\mathbf{x}_0$
g
A
$\dot{\mathbf{y}}_0$
$\int$
$\hat{\mathbf{y}}_1$
C
$\hat{\mathbf{x}}_1$
$\mathbf{u}_0$
B
$\mathbf{y}_1$
g
$\mathbf{x}_1$


$\hat{\mathbf{x}}_0$
$\hat{\mathbf{x}}_1$
$\hat{\mathbf{x}}_2$
C
C
C
$\hat{\mathbf{y}}_0$
$\hat{\mathbf{y}}_1$
$\hat{\mathbf{y}}_2$
∫
∫
∫
A
B
A
B
A
B
$\mathbf{y}_0$
$\mathbf{u}_0$
$\mathbf{u}_1$
$\mathbf{u}_2$